BTA 2023 ©

Incident Response Intro Exercises

Legend:

Input Command

Output of the previous command

Prerequisites

- Windows Operating System
- Ubuntu 22.04 Server Powered up
- Ubuntu Server on Bridged mode
- From Host OS, ssh to the Ubuntu Server

What you need to know

Linux tarballs

Linux tarballs are a common format for distributing files and directories in a Linux environment. A "tarball" refers to a collection of files and directories that have been packaged together into a single file using the `tar` (Tape Archive) command. The term "tarball" comes from the combination of "tar" and "ball," suggesting a bundle of files rolled up together. Tarballs are particularly useful for backup purposes or distributing software packages.

The `tar` Options

`c` - Create

- **Option:** `-c`
- **Description:** This option is used to create a new tar archive. It tells the `tar` command to collect files and directories into a single file (the tarball). The `-c` option needs to be followed by other options like `-f` to specify the filename of the tar archive and can be combined with compression options and the verbose option `-v` for more detailed output.

`f` - File

- **Option:** `-f`
- **Description:** This option specifies the filename of the tar archive. It is used in conjunction with other options like `-c`, `-x`, or `-t` to indicate the tarball's name that you're creating, extracting from, or viewing. The filename should immediately follow the `-f` option.

`t` - List

- **Option:** `-t`
- **Description:** This option is used to list the contents of a tar archive without extracting it. It's a way to view what's inside the tarball. When combined with the `-f` option and the archive name, it displays a list of files and directories contained within the archive.

`x` - Extract

- **Option:** `-x`
- **Description:** This option extracts files from a tar archive. When used with the `-f` option, it specifies which tarball to extract. You can also combine it with the `-v` option for verbose output, showing the files being extracted.

`v` - Verbose

- **Option:** `-v`
- **Description:** The verbose option provides detailed output during the execution of the `tar` command. When used, it lists the files being archived or extracted in real-time, giving you visibility into the process. It can be combined with options `-c`, `-t`, and `-x` for creating, listing, and extracting archives, respectively.

Using the `tar` Command

The `tar` command is used to create, maintain, modify, or extract files from a tarball. The basic syntax of the `tar` command is as follows:

```
tar [options] [archive-file] [file or directory to be archived]
```

Creating a tarball: To create a tarball, you typically use the `-c` option along with `-f` to specify the filename of the archive:

```
tar -cf archive_name.tar directory_to_compress
```

Viewing contents: To view the contents of a tarball without extracting it, you use the `-t` option:

```
tar -tf archive_name.tar
```

Extracting a tarball: To extract the contents of a tarball, the `-x` option is used along with `-f`:

```
tar -xf archive_name.tar
```

Compression

Tarballs can also be compressed using various compression algorithms. The most common compression formats used with tar are gzip (`.tar.gz` or `.tgz`), bzip2 (`.tar.bz2`), and xz (`.tar.xz`). Compression is achieved by piping the output of `tar` to a compression utility or by using `tar` with a compression option directly:

- **gzip:** `-z` option (e.g., `tar -czf archive_name.tar.gz directory_to_compress`)
- **bzip2:** `-j` option (e.g., `tar -cjf archive_name.tar.bz2 directory_to_compress`)
- **xz:** `-J` option (e.g., `tar -cJf archive_name.tar.xz directory_to_compress`)

Advantages of Using Tarballs

- **Portability:** Tarballs can be easily moved or copied between different Linux systems.
- **Efficiency:** Packaging multiple files into a single tarball can save space, especially when combined with compression.
- **Simplicity:** Tarballs make it easy to distribute and install software packages, as all necessary files are contained in a single archive.

Use Cases

- **Software distribution:** Many open-source software projects distribute their programs as compressed tarballs.
- **Backup and restoration:** System administrators often use tarballs for backup purposes because they can preserve file permissions and metadata.
- **Data transfer:** Tarballs are an efficient way to package and transfer a large number of files over the network.

Limitations

- **No built-in indexing:** Tarballs do not have an index for the files they contain, so looking inside a tarball or extracting a single file requires scanning through the entire archive, which can be time-consuming for large archives.
- **Compression is separate:** Unlike formats like ZIP, compression in tarballs is handled by separate utilities (gzip, bzip2, xz), which adds an extra step to the compression and decompression processes.

Same as zip files?

Similarities

1. **Archive Multiple Files:** Both tarballs and ZIP files are used to combine multiple files and directories into a single archive file, making it easier to manage and transport a collection of files.
2. **Compression:** Both formats can be used to compress the contents of the archives, reducing the file size for storage or transmission. However, the way compression is applied and the algorithms used can vary.
3. **Cross-Platform:** Tarballs and ZIP files can be created, modified, and extracted across different operating systems with the appropriate tools, making them useful for cross-platform file exchange.

Differences

1. **Origin and Popularity:**
   - **Tarballs** are more commonly used in Unix and Linux environments. The `.tar` extension comes from the term "Tape Archive," reflecting its origins in tape storage systems. Compression can be added using external tools, resulting in extensions like `.tar.gz` (using gzip) or `.tar.bz2` (using bzip2).
   - **ZIP files** are more prevalent in Windows environments, though they are also widely used across different platforms. ZIP files inherently support compression and do not require external tools to compress or decompress.
2. **Compression Method:**
   - **Tarballs** themselves do not compress data. Instead, the `tar` command creates an uncompressed archive, and separate compression tools like `gzip`, `bzip2`, or `xz` are used to compress the resulting tar file.
   - **ZIP files** integrate the archiving and compression into a single step, using algorithms like DEFLATE to compress the files as they are being archived.
3. **File Permissions and Metadata:**
   - **Tarballs** preserve Unix/Linux file permissions and metadata (like owner, group, and mode) by default, making them ideal for backups and software distribution in Unix/Linux environments.
   - **ZIP files** may not fully preserve Unix/Linux file permissions or special file types, which can be a limitation when transferring files between Unix/Linux systems. However, modern ZIP tools have improved in handling file permissions across different platforms.
4. **Tool Availability:**
   - **Tarballs** require the use of the `tar` command or similar archiving tools available on Unix/Linux systems (though Windows users can use tools like 7-Zip or WinRAR to work with tarballs).

- **ZIP files** can be created and extracted using built-in Windows functionality (right-click menu options) without the need for additional software. They are also easily handled on Mac and Linux systems.

5. **Use Cases:**
   - **Tarballs** are often preferred for software distribution and system backups in Unix/Linux due to their ability to handle file permissions and support for various compression algorithms.
   - **ZIP files** are commonly used for document and file sharing in more general contexts, including email attachments and downloads, due to their wide support and ease of use on Windows systems.

Tarballs are a versatile and widely used method for file distribution and backup in the Linux ecosystem. Their ability to bundle numerous files into a single archive and support for various compression algorithms make them an essential tool for system administrators and software developers.

Linux curl command

The `curl` command in Linux is a powerful and versatile tool used for transferring data to or from a server. It supports a wide range of protocols, including HTTP, HTTPS, FTP, FTPS, SCP, SFTP, TFTP, LDAP, LDAPS, DICT, FILE, IMAP, SMTP, POP3, and more. `curl` is widely used for web scripting, automation, and testing APIs. It can send requests, upload and download data, and even manage sessions. Here's a detailed look into the `curl` command:

Basic Usage

The basic syntax of the `curl` command is as follows:

```
curl [options] [URL]
```

- **URL:** Specifies the address to which `curl` will connect to perform the operation.

Common Options

- `-o` or `--output [filename]`: Saves the output to a file instead of printing it to the terminal.
- `-O`: Saves the output to a file with the same name as in the URL.
- `-d` or `--data`: Sends data to the server, making the request a POST request. It's often used when interacting with APIs or web forms.
- `-X`: Specifies the request method (e.g., `GET`, `POST`, `PUT`, `DELETE`).
- `-H` or `--header`: Adds a header to the request. It's useful for setting custom headers like `Content-Type`.
- `-u` or `--user`: Passes user authentication information. For example, for basic authentication, you might use `-u username:password`.
- `-L` or `--location`: Follows redirects. `curl` does not follow HTTP redirects by default.
- `--data-urlencode`: URL encodes the data sent in a POST request. It's useful when the data includes special characters.
- `-F` or `--form`: Submits data as if it were submitted from an HTML form, using `multipart/form-data`.
- `--cookie`: Sends cookies to the server, enabling session handling.
- `--cookie-jar`: Saves cookies from the server response, which can be used for subsequent requests.
- `-k` or `--insecure`: Allows `curl` to perform "insecure" SSL connections and transfers without verifying the SSL certificate. Useful for testing purposes on self-signed certificates.
- `-v` or `--verbose`: Provides detailed information about the request and response, useful for debugging.

Examples

Downloading a File

```
curl -O http://example.com/file.zip
```

Using Basic Authentication

```
curl -u username:password http://example.com/admin
```

Downloading Files with Redirection

```
curl -L -o filename.zip http://example.com/download
```

`curl` is a highly versatile tool essential for developers, system administrators, and power users for interacting with web services, APIs, and remote servers. Its extensive protocol support, combined with a wealth of options for customizing requests, makes it a valuable tool for automation, testing, and daily tasks involving data transfer over the network.

What is a pem file

A PEM (Privacy Enhanced Mail) identity file is a type of digital file used for storing cryptographic data like certificates, private keys, and public keys in a standardized, text-based format. The PEM format is one of the most common formats for SSL certificates and cryptographic keys and is recognized by many software applications and libraries dealing with cryptography. PEM files are typically used in secure communications and identity verification on the internet, especially in securing web traffic with SSL/TLS, authenticating servers and clients, and encrypting data.

Structure of a PEM File

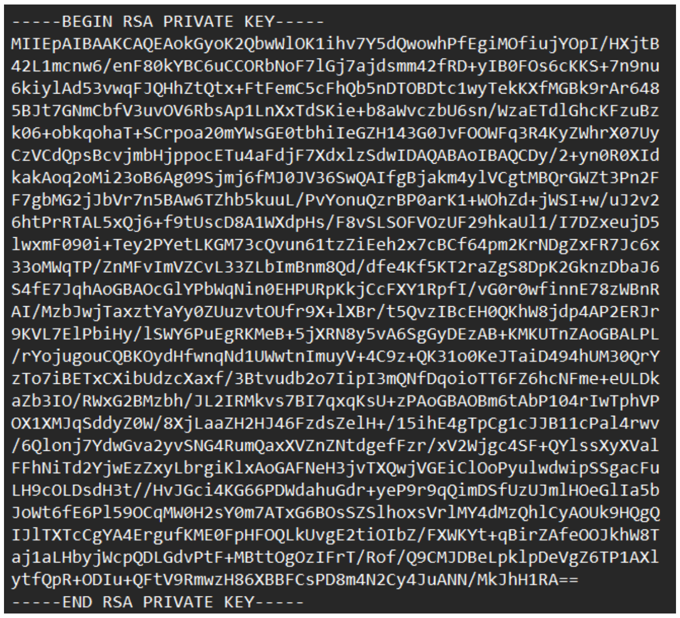
A PEM file consists of the encoded data wrapped between a header and a footer. The header and footer are easy to recognize as they contain the type of data contained in the file. For example:

-----BEGIN CERTIFICATE-----

(Base64-encoded certificate data)

-----END CERTIFICATE-----

Picture Example:



```
-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEAokGyoK2QbwWlOK1ihv7Y5dQwowhPfEgiMOfiujYOpI/HXjtB
42L1mcnw6/enF80kYBC6uCCORbNoF7lGj7ajdsmm42fRD+yIB0FOs6cKKS+7n9nu
6kiylAd53vwqFJQHhZtQtx+FtFemC5cFhQb5nDTOBDtc1wyTekKXfMGBk9rAr648
5BJt7GNmCbfV3uvOV6RbsAp1LnXxTdSKie+b8aWvczbU6sn/WzaETdlGhcKFzuBz
k06+obkqohaT+SCrpoa20mYWsGE0tbhiIeGZH143G0JvFOOWFq3R4KyZWhrX07Uy
CzVCdQpsBcvjmbHjppocETu4aFdjF7XdxlzSdwIDAQABAoIBAQCDy/2+yn0R0XId
kakAoq2oMi23oB6Ag09Sjmj6fMJ0JV36SwQAIfgBjakm4ylVCgtMBQrGWZt3Pn2F
F7gbMG2jJbVr7n5BAw6TZhb5kuuL/PvYonuQzrBP0arK1+WOhZd+jWSI+w/uJ2v2
6htPrRTAL5xQj6+f9tUscD8A1WXdpHs/F8vSLSOFVOzUF29hkaUl1/I7DZxeujD5
lwxmF090i+Tey2PYetLKGM73cQvun61tzZiEeh2x7cBCf64pm2KrNDgZxFR7Jc6x
33oMWqTP/ZnMFvImVZCvL33ZLbImBnm8Qd/dfe4Kf5KT2raZgS8DpK2GknzDbaJ6
S4fE7JqhAoGBAOcGlYPbWqNin0EHPURpKkjCcFXY1RpfI/vG0r0wfinnE78zWBnR
AI/MzbJwjTaxztYaYy0ZUuzvtOUfr9X+lXBr/t5QvzIBcEH0QKhW8jdp4AP2ERJr
9KVL7ElPbiHy/lSWY6PuEgRKMeB+5jXRN8y5vA6SgGyDEzAB+KMKUTnZAoGBALPL
/rYojugouCQBKOydHfwnqNd1UWwtnImuyV+4C9z+QK31o0KeJTaiD494hUM30QrY
zTo7iBETxCXibUdzcXaxf/3Btvudb2o7IipI3mQNfDqoioTT6FZ6hcNFme+eULDk
aZb3IO/RWxG2BMzbh/JL2IRMkvs7BI7qxqKsU+zPAoGBAOBm6tAbP104rIwTphVP
OX1XMJqSddyZ0W/8XjLaaZH2HJ46FzdsZelH+/15ihE4gTpCg1cJJB11cPal4rwv
/6Qlonj7YdwGva2yvSNG4RumQaxXVZnZNtdgefFzr/xV2Wjgc4SF+QYlssXyXVal
FFhNiTd2YjwEzZxyLbrgiKlxAoGAFNeH3jvTXQwjVGEiClOoPyulwdwipSSgacFu
LH9cOLDsdH3t//HvJGci4KG66PDWdahuGdr+yeP9r9qQimDSfUzUJmlHOeGlIa5b
JoWt6fE6Pl59OCqMW0H2sY0m7ATxG6BOsSZSlhoxsVrlMY4dMzQhlCyAOUk9HQgQ
IJlTXTcCgYA4ErgufKME0FpHFOQLkUvgE2tiOIbZ/FXWKYt+qBirZAfeOOJkhW8T
aj1aLHbyjWcpQDLGdvPtF+MBttOgOzIFrT/Rof/Q9CMJDBeLpklpDeVgZ6TP1AXl
ytfQpR+ODIu+QFtV9RmwzH86XBBFCsPD8m4N2Cy4JuANN/MkJhH1RA==
-----END RSA PRIVATE KEY-----
```

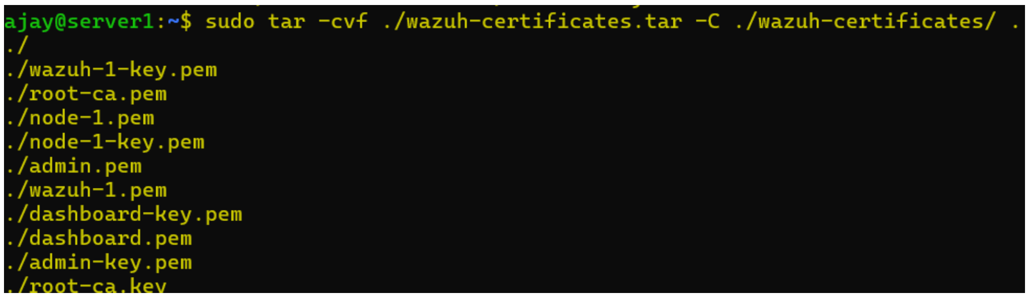
Similarly, private keys, public keys, and other cryptographic data follow the same structural pattern, with differences only in the descriptor (e.g., "CERTIFICATE", "PRIVATE KEY").

Types of Data in PEM Files

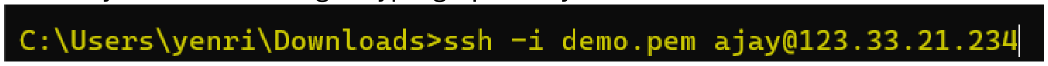
- **Certificates:** Contain public keys and identity information of a person, device, or service, signed by a trusted Certificate Authority (CA). Used in SSL/TLS for server and client authentication.
- **Private Keys:** Secret keys used in asymmetric cryptography. They should be kept confidential as they are used to decrypt data encrypted with the corresponding public key and sign data to ensure authenticity.
- **Public Keys:** Paired with private keys, public keys are used to encrypt data and verify signatures. They can be distributed openly.
- **Certificate Signing Requests (CSRs):** Requests for certificate issuance, containing information that will be included in the certificate (e.g., domain name, organization).

Usage of PEM Files

PEM files are used in various security and encryption scenarios:

- **SSL/TLS Encryption:** Web servers use PEM files to secure communications with SSL/TLS. A web server, for instance, needs a certificate (public key) and a private key to establish a secure session. These are often stored in PEM format.
    - 

```
ajay@server1:~$ sudo tar -cvf ./wazuh-certificates.tar -C ./wazuh-certificates/ .
./
./wazuh-1-key.pem
./root-ca.pem
./node-1.pem
./node-1-key.pem
./admin.pem
./wazuh-1.pem
./dashboard-key.pem
./dashboard.pem
./admin-key.pem
./root-ca.key
```

- **SSH Authentication:** PEM files can be used to authenticate SSH sessions, where a user's or host's identity is verified through cryptographic keys.
    - 

```
C:\Users\yenri\Downloads>ssh -i demo.pem ajay@123.33.21.234
```

- **Encrypting and Decrypting Data:** PEM files containing private or public keys can be used to encrypt or decrypt data, ensuring that only the intended recipient can access the information.
- **Code Signing:** Developers use PEM files containing private keys to sign their software, enabling end-users to verify the integrity and origin of the software.

Management and Security Considerations

When dealing with PEM files, especially those containing private keys, it's crucial to manage and store them securely to prevent unauthorized access. This includes:

- Setting file permissions to restrict access to sensitive PEM files.
- Using password protection or encryption for private keys.
- Regularly updating and revoking certificates and keys as necessary.

Converting Between Formats

Sometimes, it's necessary to convert PEM files to other formats (e.g., DER, PFX) depending on the requirements of different systems or software. Tools like OpenSSL allow for easy conversion between formats.

In summary, PEM identity files play a crucial role in digital security and encryption. They enable secure communication over the internet, authenticate users and devices, and ensure the integrity and confidentiality of data. Proper management and security of PEM files, especially those containing private keys, are vital to maintaining the security of digital communications and services.

I don't know what you heard about me, but I use GPG

GPG (GNU Privacy Guard) is a complete and free implementation of the OpenPGP standard, officially known as RFC 4880 (originally RFC 2440). As a part of the GNU Project, GPG is a tool for secure communication and data storage. It is widely used in the Linux environment for encryption, creating digital signatures, and managing public and private keys. GPG allows users to encrypt and sign their data and communications, features a versatile key management system, and provides access to public key directories.

Key Features of GPG

- **Encryption:** GPG can encrypt data, making it unreadable to unauthorized users. It supports both symmetric-key and public-key encryption. Symmetric-key encryption uses the same key for both encryption and decryption, while public-key encryption uses a pair of keys (a public key for encryption and a private key for decryption).
- **Digital Signatures:** GPG allows the creation of digital signatures, which verify the identity of the sender and ensure that the message or file has not been altered since it was signed. This is critical for secure communications and software distribution.

- **Key Management:** GPG provides tools for generating and managing key pairs (public and private keys). Users can create a key pair, export public keys for sharing, import others' public keys, and revoke keys if necessary.

How GPG is Used in Linux

Encrypting and Decrypting Files

- **Encrypting a file for a recipient:** To encrypt a file so that only a specific recipient can decrypt it, you use their public key. The recipient will use their private key to decrypt the file.

```
gpg --encrypt --recipient recipient@example.com file.txt
```

Decrypting a file: If you've received an encrypted file, you can decrypt it using your private key.

```
gpg --decrypt file.txt.gpg
```

Creating and Verifying Digital Signatures

- **Signing a file:** GPG allows you to sign a file with your private key. Others can verify this signature using your public key, ensuring the file's integrity and authenticity.

```
gpg --sign file.txt
```

Verifying a signature: To verify a signed file, you need the public key of the signer. GPG checks the signature against the file to ensure it hasn't been tampered with.

```
gpg --verify file.txt.sig file.txt
```

Key Management

- **Generating a new key pair:** GPG can generate a key pair (public and private keys) for you. During this process, you'll be asked to define the key size, type, and expiration.

```
gpg --gen-key
```

Exporting a public key: To share your public key with someone (or to publish it), you need to export it.

```
gpg --export --armor your@email.com > publickey.asc
```

Importing a public key: To encrypt messages for someone else or verify their signatures, you need to import their public key into your keyring.

```
gpg --import publickey.asc
```

Practical Uses of GPG

- **Email Encryption:** GPG is widely used for encrypting emails, ensuring that only the intended recipient can read the message content.
- **Software Signing:** Developers use GPG to sign their software packages. Users can verify these signatures to confirm that the packages haven't been altered and are indeed from the claimed source.
- **File Encryption:** GPG can encrypt sensitive files on your computer or in cloud storage, protecting them from unauthorized access.

Summary

GPG is a powerful tool for enhancing security in the Linux environment. Its ability to encrypt data, create digital signatures, and manage keys makes it an essential utility for anyone looking to secure their communications or verify the integrity and source of data and software. As privacy and security become increasingly important, GPG provides an accessible and robust solution for individuals and organizations alike.

Going deeper using the `sudo` command

The `sudo` command in Linux is a powerful utility that allows a permitted user to execute a command as the superuser or another user, as specified in the sudoers file. The name 'sudo' stands for "superuser do." It is used to grant administrative privileges to regular users, enabling them to perform tasks that are typically reserved for the root user.

Key Features of `sudo`

- **Security:** By using `sudo`, system administrators can limit the superuser privileges to the minimum necessary for specific tasks, reducing the risk of accidental or malicious system damage.
- **Audit Trail:** `sudo` logs all commands and attempts to use it, providing a clear audit trail of who did what. This is crucial for troubleshooting and security monitoring.
- **Flexibility:** Through the sudoers file (`/etc/sudoers`), administrators can configure detailed user privileges, specifying which commands users can run as root or other users.

Basic Usage

To use `sudo`, simply prefix the command you want to run with elevated privileges with `sudo`, like so:

```
sudo [command]
```

For example, to update the package lists on a system that uses apt:

```
sudo apt update
```

Using `sudo -i` for a Multi-Step Application Installation

The `-i` option (simulate initial login) with `sudo` starts a shell with root privileges. This can be particularly useful for running a series of commands that require superuser privileges or for executing a script or a multi-step process as the root user.

When you execute `sudo -i`, you are dropped into a root shell session, and from there, you can run multiple commands without needing to prefix each one with `sudo`. This is similar to switching to the root user with `su`, but it leverages the security and logging features of `sudo`.

Example Scenario: Multi-Step Application Installation

Suppose you need to install an application that requires several steps, including installing dependencies, copying files, and editing system configuration files. Using `sudo -i` could streamline this process:

```
sudo -i
```

<This is where you would conduct multiple commands to deploy a server or something similar<

- **Exit the Root Shell:**

Once you've completed the necessary steps, exit the root shell by typing `exit` or pressing `Ctrl+D`.

Using `sudo -i` is particularly handy when you have a script that needs to be run with root privileges, as it avoids the need to sprinkle `sudo` throughout the script or to run the script with `sudo` from the beginning.

Important Considerations

- **Security Risks:** Prolonged use of a root shell increases the risk of accidental damage to the system or misconfiguration. Always log out of the root shell when you're done to minimize this risk.
- **Sudoers Configuration:** The ability to use `sudo`, including the `-i` option, depends on the user's permissions as defined in the `/etc/sudoers` file. Ensure your user has the appropriate permissions.
- **Auditability:** Remember that actions taken from a root shell obtained via `sudo -i` are logged under the root user, which might impact the audit trail's clarity regarding who performed which actions.

`sudo` and its `-i` option are powerful tools for managing Linux systems, offering a balance between operational flexibility and security.

Exercise 1

Incident Response Steps

Task #1

List all the steps in a numbered list, and then separately explain each step in your own words.

Incident Response Phases

Task #2

List all the phases in a numbered list, and then separately explain phase step in your own words.

Incident Response Policy

Task #3

Explain what the purpose of an incident policy is in your own words.

Incident Response Plan

Task #4

Explain what an IRP is, and what purpose does it play in your own words.

Communication Plan

Task #5

Explain what an IRP is, and what purpose does it play in your own words.

Recon

Task #6

Explain what an enumeration is, and what purpose does it play in your own words.

Exfiltration

Task #7

Explain what an tunneling is, and what purpose does it play in your own words.

Communication

Task #7

Explain what a pem file is, and what purpose does it play in your own words.

Incident Response – The Dominicn Republic Incident

Submissions

- Screenshots with configurations, commands, paths, files names step by step
- Explain in your own words what is happening.

Task #1

1. Listen to this podcast on Incident Response:
    a.
    135: The D.R. Incident
    Darknet Diaries
    Omar Avilez worked in the CSIRT of the Dominican Republic when a major cyber security incident erupted. Omar walks us through what happened and the incident response...
    Jul 2023 · Played
2. Explain what your take aways are from this podcast

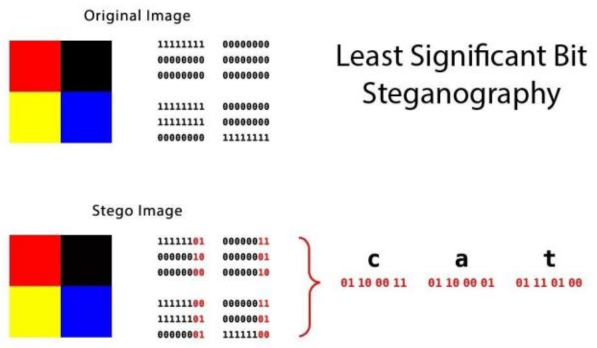
Exercise 2

Hiding in plain sight

Steganography is the practice of concealing a message or information within another non-secret file or message in a way that the existence of the hidden data is not apparent to unintended recipients. Unlike cryptography, which focuses on making the content of a message unreadable, steganography aims to hide the existence of the message itself. Common methods involve embedding information within digital images, audio files, video files, or even text. Steganography techniques include manipulating least significant bits of pixels or altering the whitespace in text documents. It's used for covert communication, digital watermarking, and protecting sensitive data.

LSB – Least Significant Bit

In steganography, the least significant bit (LSB) technique involves hiding information within the least significant bits of digital data, such as in the pixels of an image or the samples of an audio file. Since the LSBs typically contribute the least to the overall appearance or quality of the data, they can be altered without significantly affecting the perceptible quality of the file. By replacing these LSBs with the bits of the secret message, steganographers can embed hidden information within the carrier file. This method allows for the concealment of data within seemingly innocuous files, making it a popular technique in digital steganography.



https://en.wikipedia.org/wiki/Steganography

https://www.wired.com/story/steganography-hacker-lexicon/

https://www.garykessler.net/library/steganography.html

Encryption and steganography are both techniques used to protect sensitive information, but they achieve this goal in different ways.

1. **Encryption**:
   - Encryption involves encoding a message or data using an algorithm and a key to make it unreadable to anyone without the proper decryption key.
   - The focus of encryption is on making the content of the message unintelligible to unauthorized users.
   - Encrypted messages often stand out as encrypted data, and the existence of the communication itself is apparent.
2. **Steganography**:
   - Steganography, on the other hand, conceals the existence of the message itself within another file or communication channel.
   - Instead of making the content unintelligible, steganography hides the presence of the communication.
   - Steganographic techniques aim to embed secret messages within seemingly innocuous carrier files, such as images, audio files, or even text, without raising suspicion.

Using Encryption within Steganography:

While steganography typically focuses on concealing the existence of a message rather than its content, encryption can be employed within steganography to add an extra layer of security. This combination enhances the confidentiality of the hidden message by ensuring that even if the steganographic carrier is discovered, the encrypted content remains unintelligible without the decryption key.

For example, one could first encrypt a message using a strong encryption algorithm and a secret key. Then, the encrypted message can be embedded within a carrier file using steganography techniques. This way, even if someone discovers the carrier file and suspects it contains hidden information, they would still need the decryption key to uncover the actual message. This combination of encryption and steganography provides a more robust method for secure communication, obfuscation and data protection.

Steganography is used with social media platforms to transmit hidden messages by embedding them within seemingly innocuous content such as images, videos, or audio files. These platforms offer a convenient and widely accessible medium for communication, making them attractive for covert messaging.

ISIS (Islamic State of Iraq and Syria) utilized Twitter for steganographic communication by posting images with hidden messages encoded within them. These messages could contain instructions, propaganda, or other sensitive information intended for specific individuals or groups. By embedding the messages within images shared on Twitter, ISIS operatives could communicate without attracting unwanted attention from authorities or other users. This technique allowed them to bypass conventional surveillance methods and maintain a degree of anonymity while disseminating their messages.

Reference:

https://www.wired.com/story/muslimcrypt-steganography/

https://www.sans.org/white-papers/551/

https://arstechnica.com/information-technology/2015/11/isis-encrypted-communications-with-paris-attackers-french-officials-say/

Steganography – WINDOWS

Submissions

- Screenshots with configurations, commands, paths, files names step by step
- Explain in your own words what is happening.

Task #1

3. Download OpenStego from: https://www.openstego.com/

4. Download the Source File from Github: https://github.com/ajay63/BlackTowerAcademy/blob/main/maxresdefault-2003057562.jpg

Task #2 - Dependencies

If you get a dependency error, you must address that.

When you attempt to run an application on your system, the operating system expects certain dependencies to be installed in order for the application to function properly. **Dependencies are additional software components or libraries that the application relies on to execute specific functions or features.** These dependencies could include runtime libraries, frameworks, or other software packages.

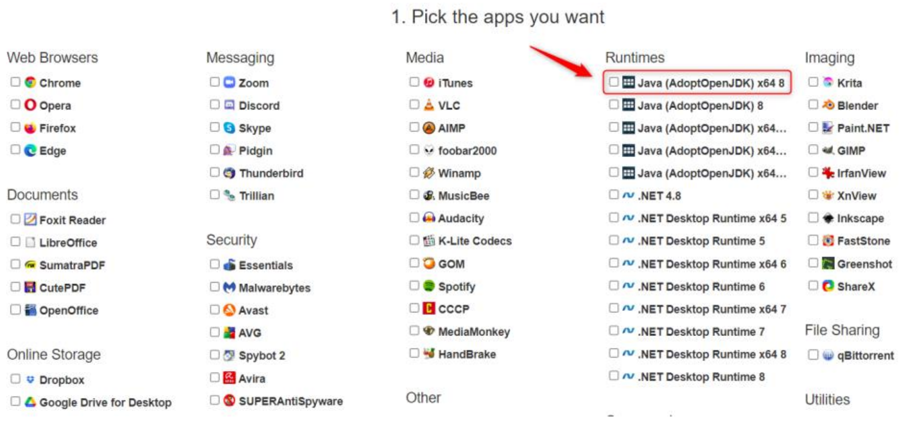
If the necessary dependencies are not installed on your system, the application may fail to run or function correctly. This failure could manifest in various ways, such as error messages, crashes, or unexpected behavior. Without the required dependencies, the application may lack the resources or functionality it needs to perform its intended tasks.

However, when an application fails to install due to missing dependencies, the installation process often provides valuable information about which dependencies are required but not found on your system. This information typically appears in error messages or logs generated during the installation attempt. These messages can help you identify which dependencies are missing and need to be installed in order for the application to run successfully.

Once you have identified the missing dependencies, you can typically install them using a package manager or by manually downloading and installing the required software packages. Once the dependencies are installed, you can attempt to run the application again, and it should now execute without issues, as it has access to all the necessary components it requires to function properly.

It is likely if OpenStego fails to run, its because it requires JAVA and you don't have it installed.

One can easily and quietly install Java from https://ninite.com/



1. Pick the apps you want

Web Browsers
- ☐ Chrome
- ☐ Opera
- ☐ Firefox
- ☐ Edge

Documents
- ☐ Foxit Reader
- ☐ LibreOffice
- ☐ SumatraPDF
- ☐ CutePDF
- ☐ OpenOffice

Online Storage
- ☐ Dropbox
- ☐ Google Drive for Desktop

Messaging
- ☐ Zoom
- ☐ Discord
- ☐ Skype
- ☐ Pidgin
- ☐ Thunderbird
- ☐ Trillian

Security
- ☐ Essentials
- ☐ Malwarebytes
- ☐ Avast
- ☐ AVG
- ☐ Spybot 2
- ☐ Avira
- ☐ SUPERAntiSpyware

Media
- ☐ iTunes
- ☐ VLC
- ☐ AIMP
- ☐ foobar2000
- ☐ Winamp
- ☐ MusicBee
- ☐ Audacity
- ☐ K-Lite Codecs
- ☐ GOM
- ☐ Spotify
- ☐ CCCP
- ☐ MediaMonkey
- ☐ HandBrake

Other

Runtimes
- ☐ Java (AdoptOpenJDK) x64 8
- ☐ Java (AdoptOpenJDK) 8
- ☐ Java (AdoptOpenJDK) x64...
- ☐ Java (AdoptOpenJDK) x64...
- ☐ Java (AdoptOpenJDK) x64...
- ☐ .NET 4.8
- ☐ .NET Desktop Runtime x64 5
- ☐ .NET Desktop Runtime 5
- ☐ .NET Desktop Runtime x64 6
- ☐ .NET Desktop Runtime 6
- ☐ .NET Desktop Runtime x64 7
- ☐ .NET Desktop Runtime 7
- ☐ .NET Desktop Runtime x64 8
- ☐ .NET Desktop Runtime 8

Imaging
- ☐ Krita
- ☐ Blender
- ☐ Paint.NET
- ☐ GIMP
- ☐ IrfanView
- ☐ XnView
- ☐ Inkscape
- ☐ FastStone
- ☐ Greenshot
- ☐ ShareX

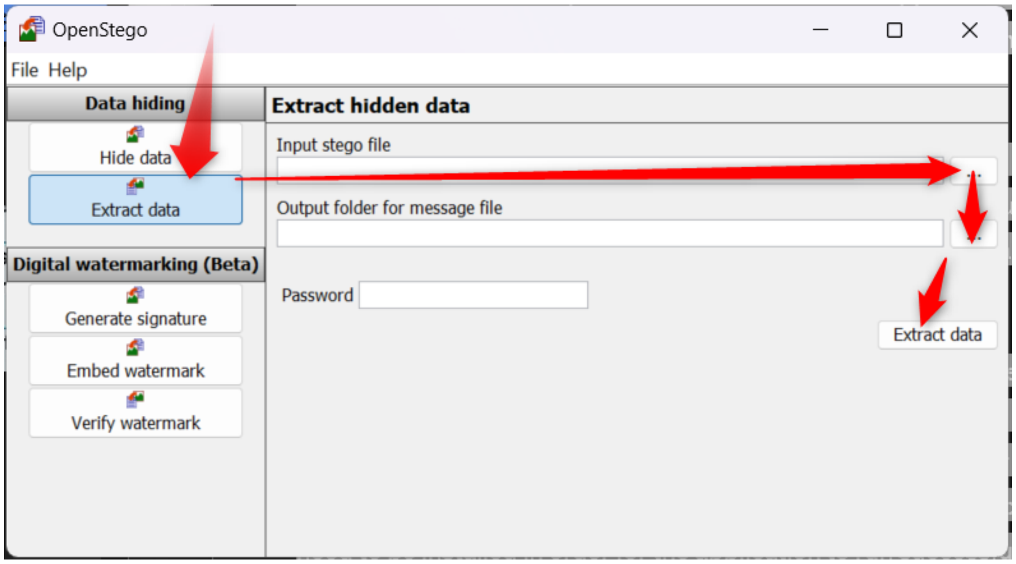
File Sharing
- ☐ qBittorrent

Utilities

Continue to follow the steps for ninite without this lab prompting you.

Task #3 – Data Extraction

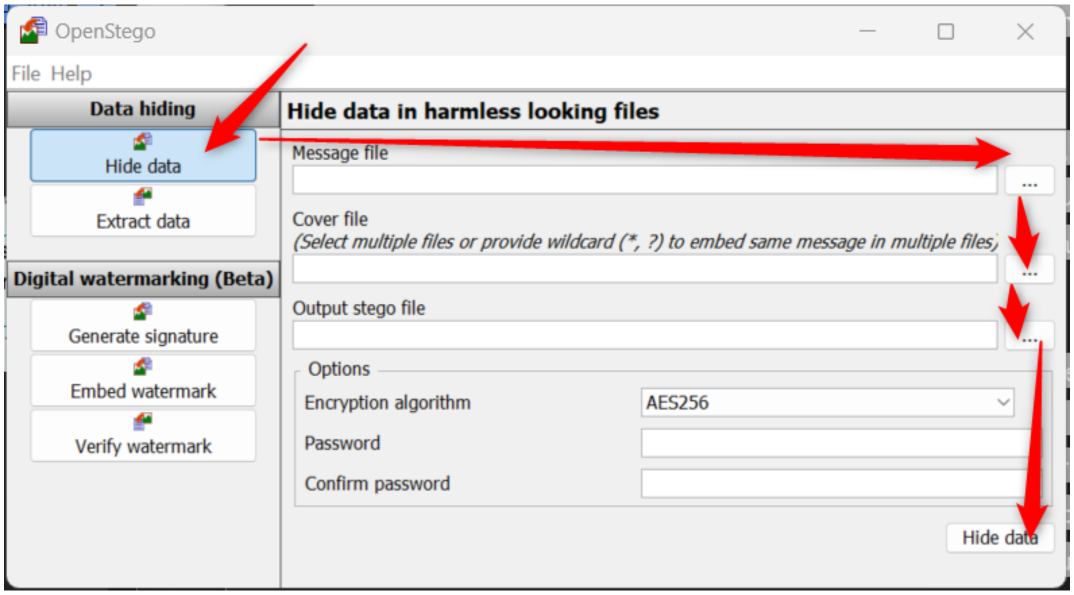
Re-run OpenStego and it should run.



5. Extract Data
6. Button, select file downloaded
7. Select folder for extraction
8. Click on EXTRACT DATA

In the output folder it will generate a file. Open the file and read the secret message hidden in the picture.

Task #3 – Implement Steganography



1. Select – Hide Data
2. Create a text file with a hidden message. (Save the file)
3. Select the text file with the hidden message. (Message File)
4. Select a funny picture you already have. (Cover File)
5. Select a location and file name for your stenographic file. (Output File)
    a. Save it as a .png file
6. Click Hide Data

Task #4 –Data Extraction

https://github.com/ajay63/BlackTowerAcademy/blob/main/evilphoto.png

Extract the Secret

Task #5 – Classmate Data Extraction

Embed a secret in an image, then share the file with a classmate, and have him read the secret message back to you.

Exercise 3

Steghide

Steghide is a popular command-line tool for hiding data or messages inside image and audio files. It is widely used in the field of information security for steganography, the practice of concealing messages or information within other non-secret data. Steghide is available on Linux systems and can be installed using package managers like apt or yum.

Task #1 – Installation

Install steghide

Task #2 – Download and extract the secret

https://github.com/ajay63/BlackTowerAcademy/blob/main/Funny-Cat-Hidden.jpg

Task #3 – Embed a secret into an image

Using steghide embed a secret into a file & share it with a classmate for them to decode.