Introduction: Linux Basics and Cryptographic Tools

CSCI 297: Ethical Hacking

About Me

- Just a guy

Course Overview

What is Hacking?

Definition: "Hacking is the practice of exploring and manipulating systems or networks in ways that deviate from their intended purposes."

Examples of Hacking Activities:

- Reverse Engineering Binaries
- Network Exploitation and Packet Spoofing
- Cryptography and Encryption Techniques
- Memory and Disk Forensics
- Circumvention of Censorship and Surveillance

Key Concept: "Being naughty is fun!"

Campus Prohibited Activities

According to university policy, the following activities are prohibited:

- Unauthorized access or disclosure of confidential information
- Misrepresentation of identity
- Tampering with computer configurations
- Impeding network traffic

For more details, refer to the University's Computing Resources Policy.

Course Objectives

In this course, we will:

- Practice Unauthorized Access
- Master Identity Misrepresentation
- Tamper with Computer Configurations
- Disrupt Network Traffic

Linux Introduction

Why Linux for Ethical Hacking?

- Stability, flexibility, and control
- Prevalence in server environments
- Strong community and resource availability

Basic Linux Commands

- Terminal introduction
- Command line navigation: `ls`, `cd`, `mkdir`, `pwd`
- Managing files and directories

Interactive Game: Bashcrawl

```
# Start Bashcrawl
cd bashcrawl
./start_game.sh
```

- Learning commands in a fun and engaging way
- Practical command-line navigation exercise
- Download Bashcrawl from: GitHub.

Introduction to Arch Linux

- Arch Linux is a lightweight and flexible Linux distribution designed for simplicity and customization.
- Follows a rolling-release model, meaning continuous updates keep your system current.
- Emphasizes simplicity and minimalism, providing users with the freedom to build their system exactly how they need it.

Pacman Package Manager

- Pacman is the package manager used in Arch Linux.
- It provides efficient package management and dependency resolution.
- Simple commands like `pacman -S package` can be used to install new packages.
- `pacman -Syu` to update the entire system.
- `pacman -Ss package` to search for a package.

Arch User Repository (AUR)

- The AUR is a community-driven repository for Arch users, providing a vast collection of user-contributed packages.
- While the AUR offers a wide range of packages, users should exercise caution and review PKGBUILD scripts for security and compatibility before installation.

Rolling Release Model

- Arch Linux follows a rolling-release model, which means no fixed release cycles.
- This model ensures that users always have access to the latest software and security patches.
- While it offers the latest software, users should be cautious during updates to avoid potential system breakages.

Viewing the Arch Linux Keyring

```
# View the Arch Linux keyring<br>
pacman-key --list-keys
```

- This command lists all the keys in the Arch Linux keyring.
- Each key is identified by its key ID, fingerprint, and associated user ID (usually an email address).
- The keyring contains keys used to sign official Arch Linux packages, ensuring their authenticity and integrity.

Understanding the Keyring

- **Keyring Use**: Holds cryptographic keys for package management and authentication.
- **Functionality**: Contains keys used to sign official packages in Arch Linux, ensuring their authenticity and integrity.
- **Management**: Users can view, manage, and verify keys to maintain system security.

Introduction to Encryption (PGP)

Importance of Pretty Good Privacy (PGP)

- **Purpose**: Encrypts data for secure communication and storage.
- **Methodology**: Uses public-key cryptography for cryptographic privacy and authentication.
- **Applications**: Widely used for digital signatures, secure email communications, and file encryption.

Introduction to Cryptography

- **Significance**: Essential in securing communications and protecting data from unauthorized access.
- **Types**: Differentiates between symmetric and asymmetric encryption.
- **Real-world Use**: Extensively applied in securing internet communications and protecting sensitive data.

What is the Web of Trust?

Concept

- **Web of Trust (WoT)**: A decentralized trust model used in PGP to establish the credibility of public key bindings.
- **Purpose**: Allows users to endorse the identity of key owners, thus extending trust.

Functionality

- **Operation**: Users sign each other's public keys to verify and trust the owner's identity.
- **Network Building**: This endorsement helps others in the network decide whom to trust.

How the Web of Trust Works

- When you sign someone's public key, you are asserting that you verify and trust the identity of the key owner.
- Others can then decide whether to trust signatures made by those keys based on their trust in you, creating a chain of trust.
- The more signatures a key has, and the more trusted those signatories are, the stronger the trust in that key becomes.

Generating PGP Keys with GnuPG

```
# Command to generate a new ECC key for signing and encrypting
gpg --full-generate-key
```

- When prompted for the kind of key you want, choose "(9) ECC (sign and encrypt)".
- Select "Curve 25519"
- Set the expiration date for the key. Typically, one to two years is suggested to balance security and maintenance.
- Enter your name, email, and an optional comment. This info will be associated with your key and public.

Managing and Uploading Your ECC Key

```
# List GPG keys for verification
gpg --list-keys
# Export your public ECC key
gpg --armor --export your-email@example.com > myecckey.asc
# Upload your key to a keyserver
gpg --keyserver keyserver.ubuntu.com --send-keys your-key-id
```

- After verifying your key details, export it in ASCII format.
- Upload the public key to a keyserver like keyserver.ubuntu.com to make it accessible.
- Sharing your public key allows others to send you encrypted messages and verify your signatures

Verifying Your Key on the Keyserver

```
# Search for your key on the keyserver
gpg --keyserver keyserver.ubuntu.com --search-keys your-email@example.com
```

- Use this command to search for your key using your email address.
- If your key appears in the search results, it confirms successful upload.
- If not, ensure you have the correct key ID and that there was no error during the upload process.
- It might take a few minutes for the key to appear due to keyserver synchronization.

Downloading a Key from the Keyserver

```
# Search for your key on the keyserver
gpg --keyserver keyserver.ubuntu.com --recv-keys C79398ABCF1852DF
```

- Replace ‘C79398ABCF1852DF‘ with the key ID you wish to download.
- This command fetches the public key and adds it to your keyring.
- Verify the key details match what you expect (e.g., owner, email).

Assigning Trust to a Key

```
# Edit the key to assign trust
gpg --edit-key C79398ABCF1852DF
```

- At the ‘gpg>‘ prompt, type ‘trust‘ to initiate the trust assignment process.
- Choose the level of trust from the options provided: 1. I do not trust 2. I do NOT trust 3. I trust marginally 4. I trust fully 5. I trust ultimately
- Confirm your selection and quit the editor with ‘save‘.

Understanding Trust Levels

- **I do not trust**: Should not be used to validate signatures.
- **I trust marginally**: Enough for signatures if multiple marginally trusted keys sign.
- **I trust fully**: Fully trusted to sign other keys and documents.
- **I trust ultimately**: This is your own key or one you equivalently trust as your own.
- Trust levels help manage and mitigate the risk of accepting fraudulent or compromised keys.

Encrypting and Signing with PGP

```
# Encrypt a file
gpg --encrypt --recipient 'name@example.com' file.txt
# Sign a file
gpg --sign file.txt
```

- Encrypting messages for secure communication
- Digital signatures for authenticity

Signing with a Specific Key

```
# Sign a document with a specific PGP key
gpg --default-key C79398ABCF1852DF --sign document.txt
```

- Replace ‘C79398ABCF1852DF‘ with the key ID you wish to use for signing.
- This command uses the specified key to sign the ‘document.txt‘.
- The ‘–default-key‘ option can be used to specify which of your keys to use for signing if you have multiple keys.

Options for Signing with GPG

```
# Sign and encrypt a document for a recipient
gpg --default-key mykeyid --sign --encrypt --recipient recipient@example.com
```

- Combining signing and encrypting: Enhances security by not only verifying the origin but also keeping the contents confidential.
- Specify the recipient's email associated with their public key for encryption.
- Always verify the recipient's key trust level and validity before sending sensitive information.

Verifying a Document's Signature

```
# Verify the signature of a document
gpg --verify document.sig document.txt
```

- Use the ‘–verify‘ option followed by the signature file and the original document.
- GnuPG checks the signature against the document to ensure that it has not been altered.
- You will receive a message indicating whether the signature is valid, who signed it, and if the signer's key is trusted.
- If the key is not already in your keyring, GnuPG will prompt that the signature cannot be verified due to a missing key.

Verifying a Signature Within a Single File

```
# Verify a signature where the document and signature are integrated
gpg --verify signedfile.txt.gpg
```

- Use the ‘–verify‘ command directly on the ‘.gpg‘ file.
- GnuPG will check the embedded signature against the content.
- The output will tell you if the signature is valid, who the signer is, and whether their key is trusted.
- This method is typical for files where the content and signature are not separated.

In-Class Activity/Homework

Today's Activity: **Secure Communication Practice**

- Each student will create a PGP key pair and sign and encrypt an email to two other students in the class.
- After completing the email exchange, each student will compile all the encrypted and decrypted files into a zip file.
- Encrypt the zip file using PGP and submit it to the course's hidden service.

Submission Link: Course's Hidden Service.

Objective: Practice secure communication techniques using PGP encryption and digital signatures.