



3차 프로젝트 보고서

≡ 태그

1. 기획

[1.1 기획 보고서](#)

[1.2 flow chart](#)

2. 내용

[2.1 다크웹 유출 정보 시스템 개발 프로젝트 일정](#)

[2.2 단계 별 프로세스](#)

[2.3 다크웹 유출정보 알림 시스템 사전 조사](#)

[2.4 다크웹 유출 알림 시스템 기능 조사](#)

[2.5 프로젝트 세부 내용](#)

[2.6 최종 결과](#)

[2.6.1 소스코드 압축파일](#)

[2.6.2 시연 영상](#)

3. 회고

[프로젝트 자체 평가 및 느낀 점](#)

[홍영창](#)

[구현모](#)

[김동진](#)

[장다솜](#)

[조운지](#)

[최승희](#)

4. 참고문헌

1. 기획

1.1 기획 보고서

▼ 팀, 다크웹 유출 알람 시스템 구축

프로젝트 명: 다크웹 유출 정보 알림 및 OSINT 수집 시스템 개발

팀명: 해커잡조

1. 개요

1.1 팀 구성원 및 담당 업무

이름	역할	담당 업무
공통진행		
홍영창	팀장	다크웹 크롤링 코드 개발 및 AWS 구축
구현모	팀원	Flask Front&Back
김동진	팀원	시스템구축 및 MongoDB 연동
장다솜	팀원	WBS 및 보고서 작성
조운지	팀원	Flask Front&Back
최승희	팀원	딥웹 크롤링 코드 개발 및 코드 통합

1.2 목적

이 프로젝트의 주된 목적은 다크웹과 같은 은밀한 사이버 공간에서 유출된 정보를 신속하게 탐지하여 정보 유출을 방지하고, 개인 및 기업의 데이터 보호를 강화하는 것입니다. 다크웹을 모니터링하고 유출된 정보를 추적하여 분석 및 보고하는 시스템을 구축하는 것을 목표로 합니다. 이를 통해 사이버 보안 위협에 대응하고 예방할 수 있는 정보를 제공하고자 하는 것입니다.

- 다크웹상의 유출된 데이터를 체계적으로 수집하고 분석하여 사이버 위협을 조기에 탐지합니다.
- 유출 정보의 출처와 유형을 파악하여 적절한 대응 전략을 수립합니다.

2. 프로젝트 필요성

다크웹에서의 정보 유출은 개인의 프라이버시 침해뿐만 아니라 기업의 중요한 비즈니스 정보 노출로 이어질 수 있습니다. 이러한 위협에 신속히 대응하기 위해 지능적인 수집 시스템이 필요합니다.

3. 주요 기능 및 스펙

- 정보 유출 신속 탐지: 다크웹을 모니터링하여 유출된 정보를 실시간으로 탐지합니다.
- 크롤링을 통한 데이터 수집: 다크웹을 크롤링하여 유출 정보를 수집합니다.
- 알람 시스템: 유출 정보가 탐지되면 slack으로 즉각 알람을 전송합니다.
- 데이터 저장 및 관리: JSON 형태로 저장되고 MongoDB에도 백업 됩니다.
- WEB 인터페이스 제공: 사용자가 쉽게 정보를 조회할 수 있도록 WEB 화면을 제공합니다.

4. 기대 효과

- 실시간 알람으로 인한 대응 시간 단축
- 유출 정보의 신속한 탐지를 통한 피해 최소화
- 데이터 유출 사전 예방을 통한 개인 및 기업 보안 강화

5. 구현 계획

본 프로젝트는 다음 단계로 구성됩니다:

1. 자료 조사 및 분석: 다크웹과 딥웹에 관한 자료조사 후 조사 리스트를 작성합니다.
2. 딥웹 다크웹 분석하기: 사이트의 구조와 특징을 분석하고 크롤링을 진행합니다.
3. 슬랙 알람 환경 구축: Slack API를 사용하여, 탐지된 정보에 대한 실시간 알람 시스템을 구축합니다.
4. 크롤링 코드 개발: 데이터 수집을 위한 크롤링 코드를 작성합니다.
5. 데이터 베이스 구축: 수집된 데이터를 저장, 검색 및 분석하기 위한 MongoDB 데이터베이스를 설계하고 구축합니다.
6. Website 구축: 사용자가 쉽게 정보를 조회하고 확인할 수 있도록 웹을 개발합니다.
7. 배포 및 모니터링: 배포하고 동작 상태를 모니터링 합니다.

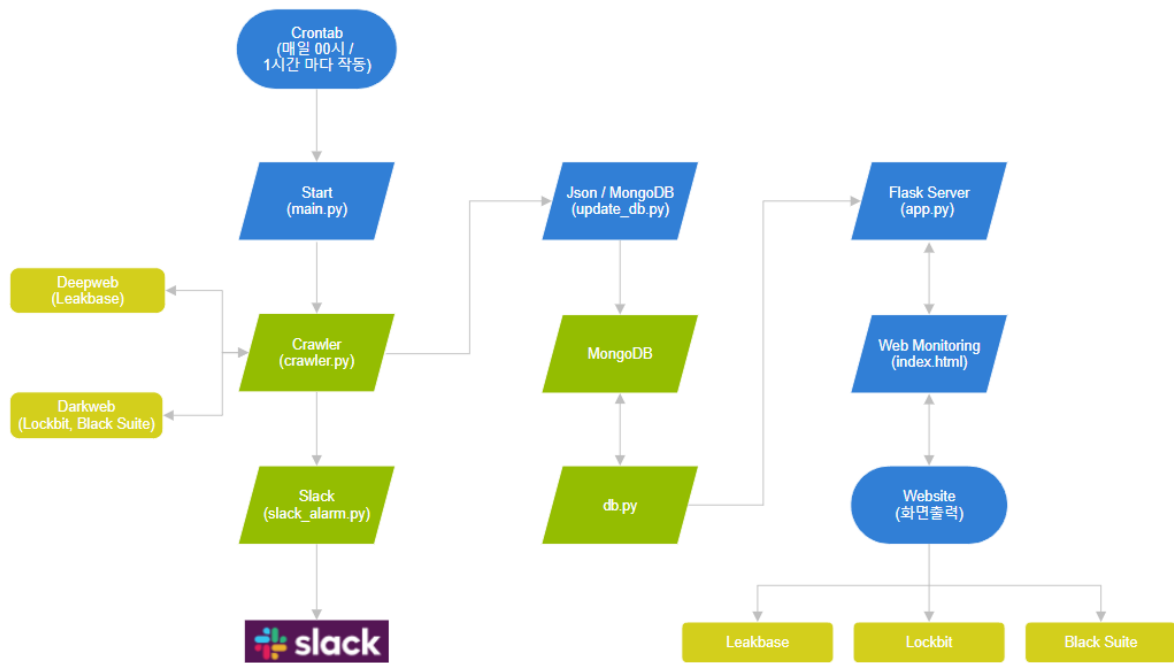
6. 기술 지원

- 개발 도구: VMware, Ubuntu, Slack, Flask, AWS
- 협업 도구: Discord
- 문서 도구: Github, Notion

7. 결론

본 시스템은 다크웹을 통한 정보 유출의 위협에 대응하는 신속하고 효과적인 솔루션을 제공합니다. 이를 통해 사용자는 데이터 유출로 인한 피해를 줄이고, 사이버 보안을 강화할 수 있을 것입니다.

1.2 flow chart



2. 내용

2.1 다크웹 유출 정보 시스템 개발 프로젝트 일정

WBS_해커잡조 (자동 저장됨).xlsx

주요 업무	세부 업무	담당	소요일	기간																									
				3월																									
				13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29									
프로젝트 기획	프로젝트 계획 수립	공통	1일																										
	다크웹 & 웹 조사	공통	1일																										
분석	다크웹 구조 파악 및 웹 크롤링	공통	3일																										
	유출 정보 알림 시스템 분석	공통	2일																										
	다크웹 & 딥웹 사이트 선정	공통	2일																										
	크롤링 개발 및 구현	영창, 승희	4일																										
	Flask Front&Back 구현	현모, 운지	6일																										
	시스템구축 및 MongoDB 연동	김동진	4일																										
	코드 통합 및 테스트	공통	3일																										
문서화 작업	WBS 및 보고서 작성	장다습	4일																										
	프로젝트 총괄, 노선 및 발표	홍영창	4일																										
	Notion & Github	공통	13일																										

2.2 단계 별 프로세스

일정	실행 내용
1주차	다크웹 & 웹 크롤링 구조 파악하기
[Day 01]	프로젝트 계획 수립
24.03.13	전회차 보고서 하나씩 맡아서 공부해보기
	다크웹, 딥웹 접속해보고 웹 크롤링 조사 및 파악
[Day 02-03]	딥, 다크웹 구조 파악
24.03.14~15	Tor 브라우저로 다크웹 접속해보기
	Leakbase(포럼), 8base(랜섬웨어) 분석해보기
	<u>hiddenwiki</u> 다크웹 주소들이 모여있음.

	한국 관련된 데이터 찾아보기
	<u>내 Gmail 주소가 다크 웹에 있는지 확인</u>
	<u>Have I Been Pwned: Check if your email has been compromised in a data breach</u>
2주차	유출 정보 알림 시스템 개발
[Day 04-05]	크롤링 및 slack, telegram 알림 설정
24.03.18~19	슬랙, 텔레그램을 통한 알림 연동 구현 > 성공
	Beautifulsoup4 와 selenium을 이용한 웹 크롤링 테스트 실패 및 구조 파악
	leakbase 크롤링 시도해서 데이터 가져와보기
[Day 06]	유출 정보 알림 시스템 분석
24.03.19~20	다크웹 유출정보 알림 시스템 사전 조사
	다크웹의 특정 게시판이나 글을 상대로 스크래핑 해보기
	스크래핑한 글들을 대상으로 텔레그램 or 슬랙 연동시켜보기
[Day 07]	다크웹 & 딥웹 사이트 선정
24.03.21	<u>lockbit3, leakbase</u> 선정
	lockbit3 크롤링(캡처 , title) 성공
	leakbase 제목, 작성자, 시간 json형태로 저장
	leakbase 크롤링 진행
[Day 08]	코드 작성 및 개발
24.03.22	모듈화, 프론트 엔드 참고
	코드 수정사항 디코에 공유
3주차	알림 시스템 개발 및 테스트
[Day 09~10]	슬랙 연동 및 웹사이트 구현
24.03.25~03.26	web 서버생성 후 flask백엔드 및 front-end 웹페이지 구상 및 만들기
	락빗, 릭베이스 한 페이지에 보이게 사이트 구상하자.
	index파일 만들어봄 DB연동은 아직
	플라스크-html 만들, 몽고디비 연동
	구름 ide 크레딧 및 기술적 문제로 > aws 배포로 변경
[Day 11~12]	프로그램 개발 완성 및 테스트
24.03.27~03.28	leakbase 접속 안됨(지금은 접속 됨) > <u>블랙수트</u> 하나 더 추가
	코드 통합 및 테스트
	보고서 작성
[Day 13]	제출 및 발표
24.03.29	영창님 - 제출 및 발표

▼ 2.3 다크웹 유출정보 알림 시스템 사전 조사

이름	링크	포럼/랜섬	한국관련 정보
릭베이스	https://leakbase.io/forums/apbchucky	딥 / 포럼	Y
블랙수트	http://weg7sdx54bevnvulapqu6bpzwwztryeflg3s23tegbmnhkbpqz637f2yd.onion/	다크	
해튼		다크 / 회원제	
블랙캣	http://alphvmmmm27o3abo3r2mlmjrpdmzle3rykajqc5xsjZi7ejksbpsa36ad.onion	다크 / 랜섬	검거됨
8BASE	http://xb6q2aggyclmcrjtbjendcnwpmmbosqaugxsqb4nx6cmod3emy7sad.onion/	다크 / 랜섬	Y
DOUNT LEAKS	http://sbc2zv2qnz5vubwtx3aobfpkeao6l4igjegm3xx7tk5suqhjkp5jxtqd.onion/	다크	
bianian	http://bianlianlbc5an4kgnay3opdemgcryg2kpfcbgczopmm3dnbz3uaunad.onion/	다크 / 랜섬	
어비스	http://3ev4metjrohtdpshsqlkrqcmxq6zu3d7obrdhgjpy5jpb7whmlfggd.onion/	다크	
아키라	https://akiral2iz6a7qgd3ayp3l6yub7xx2uep76idk3u2kollpj5z3z636bad.onion/	다크	
PLAY	http://weg7sdx54bevnvulapqu6bpzwwztryeflg3s23tegbmnhkbpqz637f2yd.onion/	다크	

락비트3	lockbit3753ekiocy05epmpy6klmejchjtzddoekjint6mu3qh4de2id.onion	다크	
BreachForums	http://breachedu76kdyavc6szj6ppbpifqoz3pgrk3zw57my4vybgblpfeayd.onion/	다크 / 포럼	
	breachforums.is	딥 / 포럼	
어니언	http://omegalock5zxwbhswbisc42o2q2i54vdulyvtqqbudqousisjgc7j7yd.onion/	다크	

▼ 2.4 다크웹 유출 알림 시스템 기능 조사

분석

S2W lab

- **Active Threat and Vulnerability Management**

다크웹에서 활동 중인 랜섬웨어 정보 모니터링

최신 취약점/연관 침해지표(IoC)/야라룰&스노트를 제공

공격 표면 모니터링(Attack Surface Monitoring)

위협 그룹 및 악성코드 관련 인텔리전스 제공

- **Data Breach Detection**

다크웹/언더그라운드 커뮤니티 내 기업 내부 자료 유출 탐지

카드데이터/은행 계좌 정보 등 금융 정보 유출 탐지

임직원 계정 정보 유출 탐지

유출된 중요 자산 키워드 알림 서비스 제공

- **Dark Web Monitoring**

딥/다크웹 검색 엔진 제공

다양한 식별자 검색 및 교차 분석 검색 가능

고객사 요청에 따른 맞춤형 사이트 정보 수집

다크웹 동향 리포트 제공

NSHC

다크웹 모니터링관련 위협 정보를 실시간으로 전달

StealthMole의 사고 모니터링 모듈은 다크 웹에 게시된 위협 및 보안 사고에 대한 실시간 데이터의 지속적인 스트림을 제공합니다. 고급 AI 기반 웹 크롤링 기술을 기반으로 하는 사고 모니터링 모듈은 사용자에게 랜섬웨어 공격, 데이터 침해 또는 국가 표적 사이버 위협의 첫 번째 징후를 경고합니다.

- **랜섬웨어 모니터링**

StealthMole은 새로운 랜섬웨어 캠페인이 있는지 다크 웹을 적극적으로 모니터링하여 위협에 대해 즉각적인 조치를 취할 수 있도록 실시간 경고를 제공합니다. 적용 범위에는 광범위한 랜섬웨어 웹사이트, 블로그 및 기타 다크 웹의 숨겨진 소스가 포함되어 모든 근거를 포괄합니다.

(주요정보: 피해자/랜섬웨어 갱/탐지 날짜/랜섬웨어 URL/피해자 사이트/피해자 국가/부문)

- **유출된 데이터 모니터링**

StealthMole은 침해 포럼, 마켓플레이스, 소셜 미디어 채널 및 다양한 공개 및 비공개 소스를 통해 다크 웹을 포괄적으로 크롤링하여 데이터 유출 또는 데이터 침해 사고를 구체적으로 탐지합니다.

(주요정보: 데이터 유출 제목/불량 행위자/탐지 날짜/소스 URL)

- **정부 모니터링**

StealthMole은 주 정부 및 기관에 초점을 맞춰 딥 웹과 다크 웹을 검색하여 표적 공격, 데이터 침해 및 기타 악의적 활동에 대한 지표를 찾습니다. 실시간 경고를 통해 국가 안보에 대한 위협이 발생하는 즉시 정부에 경고가 전달됩니다.

(주요정보: 데이터 유출 제목/불량 행위자/탐지 날짜/소스 URL)

제로다크웹

- 정보 유출 감지 및 모니터링

제로다크웹 서비스는 매일 다크웹에서 정보유출이 발생하는지 모니터링합니다.

이를 위해 첨단 보안 기술과 알고리즘을 활용하여 다크웹을 스캔하고, 사용자의 도메인과 관련된 정보 유출 여부를 실시간으로 감지합니다

- 정기 리포트 발송

매주 정기적으로 서비스는 사용자에게 리포트를 발송합니다. 이 리포트에는 해당 주 동안 다크웹에서 감지된 정보유출 사례 및 서비스의 작동 상태에 대한 요약 정보가 포함됩니다. 이를 통해 사용자는 주기적으로 서비스의 성과를 확인할 수 있습니다.

- 긴급 알림 메일 발송

정보유출이 감지되면 제로다크웹 서비스는 사용자에게 긴급 알림 메일을 발송합니다. 이 메일은 사용자가 즉각적으로 정보유출 사실을 알 수 있도록 돕습니다.

쿠크

- 사건 조사를 위한 아바타 생성 및 관리

CHE는 다양한 플랫폼에서 정보를 수집하고 사이버 범죄에 연루된 것으로 판단되는 특정 용의자에게 접근하기 위한 가상 인물인 “아바타”를 생성하고 관리하는 기능을 제공하여, 범죄 활동 추적을 위한 정확한 정보와 증거 수집을 지원합니다. **CHE**의 아바타 기능은 각 범죄 유형에 맞춰 생성할 수 있도록 별도의 전문 교육과 함께 제공됩니다.

계정생성 > 아바타 관리 연동 > 아바타 지정 > 수집 활동

- 데이터 상관 관계 분석 및 위치 추적

다양한 수집 데이터 중 지정된 타겟, 키워드, 관련 인물, 활동, 행위, 이벤트 등의 상관 관계를 실시간으로 분석하고 시각화 하여 제공합니다. 이러한 다양한 수집 데이터의 상관 관계 분석을 통해 범죄자의 행위들을 수집하고, 프로파일을 수행할 수 있습니다. **CHE**를 통해 기존에는 수행할 수 없었던 게스트 및 고스트 계정의 추적에 정확도를 높일 수 있고 실제 계정 사용자를 정확하게 추적할 수 있는 확률을 높입니다.

NordVPN

- 연중무휴 24시간 보안

다크 웹 모니터링은 각종 다크 웹 포럼과 사이트에서 사용자의 NordVPN 이메일 주소와 연결된 자격 증명이 발견되는지 지속적으로 검색합니다. 보고할 내용이 없는 동안에는 백그라운드에서 조용히 실행됩니다.

- 원클릭 보호

사용자 데이터 유출에 대한 뉴스를 접할 때마다 신경 써서 다크웹 모니터링을 켜 필요가 없습니다. 한 번 활성화해두기만 하면 이 도구는 사용자의 계정을 지속적으로 보호해 줍니다.

- 즉각적인 위협 알림

다크 웹 모니터링은 다크 웹 페이지에서 사용자의 인증 정보를 발견하는 즉시 알려줍니다. 유출이 발생한 서비스 및 관련된 모든 관련 정보가 제공됩니다.

정리

이 다섯 기업의 다크 웹 모니터링 서비스는 모두 사이버 보안 위협에 대응하여 개인 및 기업의 정보를 보호하는 데 초점을 맞추고 있으나, 제공하는 기능과 서비스의 세부적인 측면에서 공통점과 차이점이 있다.

▼ 공통점:

1. **다크웹 및 딥웹 모니터링:** 모든 서비스는 다크웹 및 관련 사이버 공간에서의 활동 모니터링을 제공하며, 랜섬웨어 정보, 데이터 유출 탐지, 공격 표면 모니터링 등에 초점을 맞추고 있다.
2. **실시간 알림 기능:** 유출이나 위협이 감지되면 실시간으로 알림을 제공하는 기능을 포함하여 사용자나 기업이 즉각적으로 대응할 수 있도록 지원한다.

3. **사용자 지정 가능성:** 대부분의 서비스는 고객의 요구에 따라 맞춤형 모니터링 옵션을 제공하여 특정 요구 사항에 부합하는 정보 수집 및 분석이 가능하다.

▼ 차이점:

1. **S2W lab**

- 주로 활동 중인 랜섬웨어 정보 모니터링과 취약점 관리에 중점을 둔다.
- 공격 표면 모니터링과 위협 그룹 관련 인텔리전스를 제공한다.
- 다양한 언어를 지원하며, 전문가들의 경험을 바탕으로 운영된다.

2. **NSHC:**

- 랜섬웨어 및 데이터 유출에 대한 실시간 모니터링과 경고 시스템을 강조한다.
- 고급 AI 기술을 활용하여 사이버 위협을 감지하고 대응한다.
- 정부 및 기관에 초점을 맞춘 모니터링을 제공하여 국가 안보에 대한 위험도 감지한다.

3. **제로다크웹:**

- 정보 유출 감지 및 정기 리포트 발송에 초점을 맞추고 있다.
- 긴급 알림 메일 서비스를 제공한다.

4. **쿤택:**

- 가상 인물인 아바타 생성 및 관리 기능을 제공한다.
- 데이터 상관계 분석과 위치 추적을 통해 범죄자의 행위를 추적하고 분석한다.
- 게스트 및 고스트 계정의 추적에 정확도를 높이고 실제 계정 사용자를 추적할 수 있는 확률을 높인다.

5. **NordVPN:**

- 다크웹 모니터링을 통해 사용자의 이메일 주소와 연결된 유출을 지속적으로 검색한다.
- 간편한 원클릭 보호 기능을 강조한다.

S2W lab, NSHC, 제로다크웹은 기업과 조직을 대상으로 하는 서비스에 중점을 두는 반면, **NordVPN**은 개인 사용자를 위한 서비스에 더 초점을 맞추고 있다.

쿤택은 사이버 범죄 조사 기관이나 법 집행 기관을 위한 전문화된 서비스를 제공한다.

▼ 2.5 프로젝트 세부 내용

기술스택

구분	기술/도구	설명
데이터 수집	BeautifulSoup	정적 웹 페이지에서 데이터를 추출할 때 사용. HTML과 XML문서를 파싱하여 추출한다.
	Selenium	웹 브라우저 자동화를 통해 실시간 데이터 크롤링에 사용된다. 특히, socks5 프록시와 토르 브라우저를 이용해 다크웹에서도 데이터 수집이 가능하며, 크롬 드라이버를 통한 제어가 필요하다
데이터 저장	JSON	크롤링된 데이터를 구조화된 형식으로 저장하기 위해 사용. 데이터 교환 및 저장에 JSON 형식을 활용
	MongoDB	MongoDB는 문서 지향적 NoSQL 데이터베이스로, 대규모 데이터셋을 효율적으로 처리할 수 있는 유연성과 확장성을 제공한다. JSON과 유사한 BSON 형식을 사용해 데이터를 저장하며, 스키마가 없어 다양한 형태의 데이터를 유연하게 저장할 수 있다.
알림 시스템	Slack API	크롤링 작업 완료 시 사용자에게 알림 메시지를 전송하기 위해 사용한다.
프론트엔드 개발	index.html	웹 인터페이스의 구조를 정의하는데 사용되는 HTML 파일
백엔드 개발	Flask	웹 애플리케이션의 백엔드 개발을 위한 경량의 Python 웹 프레임워크. 빠른 개발과 쉬운 통합을 지원한다.
인프라 구축	AWS	클라우드 기반 인프라 서비스를 제공하는 플랫폼으로, 확장성과 유연성을 갖춘 다양한 컴퓨팅 자원과 서비스를 제공한다.

파일 구조

```
hackerzobjo/
├─ app.py
├─ config.py
├─ crawler.py
├─ db.py
├─ main.py
├─ slack_alarm.py
├─ update_db.py
├─ webdriver_setting.py
├─ templates/
│   └─ index.html
├─ data/
│   └─ leakbase_data.json
│   └─ lockbit_data.json
│   └─ blacksuit_data.json
└─ __pycache__
```

대상 사이트

Dark Web	Ransomware Blog
Lockbit	http://lockbitapt2yfbt7lchxejug47kmqvgqxvvjpgkmevv4l3azl3gy6pyd.onion/
BlackSuit	http://weg7sdx54bevnvulapqu6bpzwztryeflg3s23tegbmnhkbpqz637f2yd.onion/
Deep Web	Leak Forum
LeakBase	https://leakbase.io

cron 작업 스케줄러

```
0 * * * * /usr/bin/python3 /home/ubuntu/crawler/main.py && /usr/bin/python3 /home/ubuntu/c
```

cron 작업 스케줄러를 통해 `crontab -e` 명령어를 통해 시스템에 설정된 스케줄에 따라 매 시간마다 `/home/ubuntu/crawler/main.py` 스크립트를 실행하고, 그 작업이 성공적으로 완료되면 이어서 `/home/ubuntu/crawler/update_db.py` 스크립트를 실행한다.

위 과정을 통해 지정된 시간에 자동으로 해당 Python 스크립트가 실행되도록 한다.

crawler.py

▼ crawler.py 코드

```
import pytz
import requests
from datetime import datetime
from urllib.parse import urljoin
from bs4 import BeautifulSoup
from webdriver_setting import open_driver
# from storage import check_posts
from selenium.common.exceptions import TimeoutException, WebDriverException
from requests.exceptions import RequestException
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
```



```

from selenium.webdriver.common.by import By
from config import LEAKBASE_FILE_PATH, LOCKBIT_FILE_PATH, BLACKSUIT_FILE_PATH
import json
from slack_alarm import slack_alarm

# 이전 게시글 데이터 불러오기
def load_previous_posts(site):
    if site == 'leakbase':
        file_path = LEAKBASE_FILE_PATH
    elif site == 'lockbit':
        file_path = LOCKBIT_FILE_PATH
    elif site == 'blacksuit':
        file_path = BLACKSUIT_FILE_PATH

    try:
        with open(file_path, 'r', encoding='utf-8') as file:
            previous_posts = json.load(file)
    except FileNotFoundError:
        previous_posts = []
    return previous_posts

# 새로운 게시글 확인 및 처리
def check_posts(new_posts, site):
    previous_posts = load_previous_posts(site)
    if not previous_posts:
        for post in new_posts:
            slack_alarm(post, site)
        update_file(new_posts, site)
        return

    if site == 'leakbase':
        previous_urls = set(post['url'] for post in previous_posts)
        new_posts_found = [post for post in new_posts if post['url'] not in previous_urls]
    elif site == 'lockbit':
        previous_urls = set(post['title'] for post in previous_posts)
        new_posts_found = [post for post in new_posts if post['title'] not in previous_urls]
    elif site == 'blacksuit':
        previous_urls = set(post['title'] for post in previous_posts)
        new_posts_found = [post for post in new_posts if post['title'] not in previous_urls]
    else:
        print('Error')
        return

    if new_posts_found:
        for post in reversed(new_posts_found): # 슬랙 알림 과거-최신순으로 보내도록
            slack_alarm(post, site)
        updated_posts = new_posts_found + previous_posts
        update_file(updated_posts, site)
    else:
        update_file(previous_posts, site)

# 파일 업데이트
def update_file(posts, site):
    # crawled_time = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
    if site == 'leakbase':
        file_path = LEAKBASE_FILE_PATH

```

```

elif site == 'lockbit':
    file_path = LOCKBIT_FILE_PATH
elif site == 'blacksuit':
    file_path = BLACKSUIT_FILE_PATH
with open(file_path, 'w', encoding='utf-8') as file:
    json.dump(posts, file, ensure_ascii=False, indent=4)

    print(f'json file path: {file_path} updated.')

# URL 페이지를 가져와 파싱, 타임아웃 발생시 브라우저 종료하고 예외 발생
def fetch_and_parse_html(driver, url, wait_element=None):
    try:
        driver.get(url)
        if wait_element:
            WebDriverWait(driver, 20).until(
                EC.presence_of_element_located((By.CSS_SELECTOR, wait_element))
            )
        return BeautifulSoup(driver.page_source, 'html.parser')
    except TimeoutException:
        print(f"타임아웃 에러 발생 URL: {url}. 다시 시도해주시길 바랍니다.")
        raise TimeoutException(f"Failed to load {url}, browser closed.")

# leakbase
def fetch_leakbase_data():
    site = 'leakbase'
    leakbase_url = 'https://leakbase.io/'

    try:
        response = requests.get(leakbase_url)
        response.raise_for_status()
        html = response.text
        soup = BeautifulSoup(html, 'html.parser')

        # 게시물 정보 추출
        post_elements = soup.select('div._xgtIstatistik-satir--konu')
        time_elements = soup.select('time.structItem-latestDate')
        forum_elements = [a['title'] for a in soup.select('._xgtIstatistik-satir--hucre

posts = []

for post_element, time_element, forum in zip(post_elements, time_elements, foru
    # 작성자 정보
    author = post_element.get('data-author', 'no author')
    # url
    title_tag = post_element.find('a', attrs={'data-preview-url': True})
    # 게시물 시간정보
    datetime_str = time_element.get('datetime', 'no time info')
    utc_datetime = datetime.strptime(datetime_str, '%Y-%m-%dT%H:%M:%S%z')
    korea_datetime_str = utc_datetime.astimezone(pytz.timezone('Asia/Seoul')).s
    time_text = time_element.get_text().strip()

    # 게시물 주소
    if title_tag:
        full_url = urljoin(leakbase_url, title_tag['href'])
        title = title_tag.get_text().strip()

        post_data = {

```

```

        'title': title,
        'upload time': korea_datetime_str,
        'time_text': time_text,
        'author': author,
        'url': full_url
    }
    posts.append(post_data)

    # 새로운 게시물 확인 및 처리
    check_posts(posts, site)
except RequestException as e:
    print(f'Request to {leakbase_url} failed: {e}')

# lockbit
def fetch_lockbit_data():
    site = 'lockbit'
    driver = open_driver()
    URL = "http://lockbit7ouvrsgdtojeoj5hvu6bljqthitekwpdy3b6y62ixtsu5jqd.onion"

    try:
        soup = fetch_and_parse_html(driver, URL)

        posts = []

        post_container = soup.find('div', class_='post-big-list')
        if post_container:
            post_blocks = post_container.find_all('a', class_='post-block')

            for post in post_blocks:
                title = post.find('div', class_='post-title').text.strip()
                post_text = post.find('div', class_='post-block-text').text.strip()
                updated_date = post.find('div', class_='updated-post-date').text.strip()
                days_element = post.find('span', class_='days')
                post_url = post.get('href')
                full_url = urljoin(URL, post_url)
                if days_element:
                    days = days_element.text.strip()
                else:
                    days = 'published'

                post_data = {
                    'title': title,
                    'post_text': post_text,
                    'upload time': updated_date,
                    'days': days,
                    'url': full_url
                }
                posts.append(post_data)

            check_posts(posts, site)
        except (TimeoutException, WebDriverException) as e:
            print(f'An error occurred: {e}')
        finally:
            driver.quit()

def fetch_blacksuit_data():
    site = 'blacksuit'
    driver = open_driver()

```

```

URL = "http://weg7sdx54bevnvulapqu6bpzwttryeflq3s23tegbmnhkbpqz637f2yd.onion"
current_page_number = 1
posts = [] # 포스트 리스트를 루프 밖에 위치

try:
    while True:
        soup = fetch_and_parse_html(driver, f"{URL}/?page={current_page_number}")
        post_container = soup.find('main')

        if post_container:
            post_blocks = post_container.find_all('div', class_='card')

            for post in post_blocks:
                try:
                    title = post.find('div', class_='url').find('a')['href'].strip()
                    post_text = post.find('div', class_='text').text.strip()
                    links = []
                    link_div = post.find('div', class_='links')

                    if link_div:
                        for lk in link_div.find_all('a'):
                            url_title = lk.text.strip()
                            url_link = lk['href']

                            links.append({
                                'url-title': url_title,
                                'url-link': url_link,
                            })

                    post_data = {
                        'title': title,
                        'post-text': post_text,
                        'url': links,
                    }

                    posts.append(post_data) # 포스트 데이터를 포스트 리스트에 추가

                except (TimeoutException, WebDriverException) as e:
                    print(f'An error occurred: {e}')

            pagination_links = driver.find_elements(By.CSS_SELECTOR, ".pagination a")
            if pagination_links and current_page_number < int(pagination_links[-1].text):
                current_page_number += 1
            else:
                break
            check_posts(posts, site)

        except (TimeoutException, WebDriverException) as e:
            print(f'An error occurred: {e}')

    finally:
        driver.quit()

```

cron 스케줄러를 통해 main.py가 실행되면 crawler.py가 실행된다.

crawler.py에서는 Tor Proxy socks5 9050을 이용하려면 토르 브라우저를 이용해야 되기 때문에 selenium의 chromedriver를 이용한다.

BeautifulSoup과 selenium을 활용하여 Leakbase, Lockbit, 그리고 Blacksuit 사이트에서 데이터를 추출하고 기존 Json 폴더가 있는지 먼저 확인하고 없으면 slack_alarm() 함수가 실행되어 알람을 뿌려주고 Json파일로 저장한다.

만약 저장된 기존 Json파일이 있을 경우 이전 게시물들과 비교해서 새로운 게시물을 발견하면 slack을 통해 알림을 전송한다.

update.py

▼ update.py 코드

```
from pymongo import MongoClient
import json

# MongoDB 설정
MONGO_HOST = 'localhost'
MONGO_PORT = 27017
LEAKBASE_DB_NAME = 'leakbase'
LOCKBIT_DB_NAME = 'lockbit'
BLACKSUIT_DB_NAME = 'blacksuit'
COLLECTION_NAME = 'posts'

def update_database(file_path, database_name, main_field):
    # MongoDB 클라이언트 생성
    client = MongoClient(MONGO_HOST, MONGO_PORT)
    db = client[database_name]
    collection = db[COLLECTION_NAME]

    # JSON 파일 로드
    with open(file_path, 'r') as file:
        data = json.load(file)

    # MongoDB에 데이터 업데이트
    for item in data:
        # MongoDB에서 동일한 'url'을 가진 문서를 찾아 업데이트하거나, 존재하지 않는 경우 새로운 문서 추가
        collection.update_one({main_field: item[main_field]}, {'$set': item}, upsert=True)

    print("Database has been updated.")

# 스크립트 실행
if __name__ == "__main__":
    update_database('/home/ubuntu/crawler/data/leakbase_data.json', LEAKBASE_DB_NAME, 'url')
    update_database('/home/ubuntu/crawler/data/lockbit_data.json', LOCKBIT_DB_NAME, 'title')
    update_database('/home/ubuntu/crawler/data/blacksuit_data.json', BLACKSUIT_DB_NAME, 'url')
```

cron 스케줄러를 통해 두번째로 실행된다.

MongoDB를 활용하여 크롤링한 데이터를 데이터베이스에 저장하거나 업데이트하는 작업을 자동화한다.

각각의 다크웹 사이트(Leakbase, Lockbit, Blacksuit)에 대한 데이터는 JSON 파일로부터 읽어들이지며, 주요 필드(`url` 또는 `title`)를 기준으로 이를 MongoDB의 해당 데이터베이스와 컬렉션에 저장한다.

그리고

`update_one` 함수를 통해 MongoDB에서 동일한 'url'을 가진 json데이터를 찾아 업데이트 하거나 존재하지 않는 경우 새로운 데이터를 추가한다.

app.py

▼ app.py 코드

```
from flask import Flask, render_template
from db import get_leakbase_posts, get_lockbit_posts, get_blacksuit_posts
```

```

app = Flask(__name__)

@app.route('/')
def index():
    # MongoDB에서 데이터 가져오기
    leakbase_posts = get_leakbase_posts()
    lockbit_posts = get_lockbit_posts()
    blacksuit_posts = get_blacksuit_posts()
    # 템플릿에 데이터 전달
    return render_template('index.html', leakbase_posts=leakbase_posts, lockbit_posts=1

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000, debug=True)

```

cron 스케줄러를 통해 main.py와 update.py가 실행되면 Flask 웹 애플리케이션을 설정하고 구동한다.

MongoDB에서 Leakbase, Lockbit, Blacksuit 사이트의 데이터를 가져온 다음, 이 데이터를 `index.html` 템플릿으로 전달하여 사용자에게 보여준다.

2.6 최종 결과

2.6.1 소스코드 압축파일

[hackerjobjo.zip](#)

2.6.2 시연 영상

https://prod-files-secure.s3.us-west-2.amazonaws.com/601f866a-fc0b-4e79-9803-30a518d349f4/6918aa3f-464f-468e-aa04-58f458705035/hackerjobjo_crawler.mp4

3. 회고

프로젝트 자체 평가 및 느낀 점

홍영창

웹 크롤링은 이름만 들어봤지 실제로 어떻게 작동하는지에 대해선 자세히 알지 못했는데 이번 다크웹 유출 알림 시스템 프로젝트를 진행하며 BeautifulSoup와 Selenium을 사용하며 복잡한 웹 크롤링을 해냈고 이에 대해 이해할 수 있었다.

특히, 다크웹 사이트를 크롤링하기 위해 필요했던 Tor와 Socks5 프로토콜에 대해 처음 배우며 어려움을 겪었지만, 팀원들의 도움으로 어느정도 이해하고 나서는 크롬 웹 브라우저가 켜지지 않도록 headless를 주면서 크롤링하는 방법을 터득했다.

AWS구축도 처음으로 도전하게 되는데 걱정이 많았지만 하나하나 해결해가며 성공적으로 서버를 배포하게되어 클라우드 컴퓨팅에 대한 이해도를 높인 것 같아 좋은 경험이었다.

프로젝트 마지막 단계에서는 MongoDB와 Flask, HTML을 활용하여 웹 모니터링 시스템을 구축했는데 이 과정에서 이전에 배우고 사용했던 경험을 다시 활용해서 프로젝트에 반영할 수 있어 뿌듯했다.

구현모

다크웹에 관심만 있고 접속해본 적은 없었는데 이번 기회를 통해 경험해 만족한다.

크롤링과 슬랙 봇 등 구조와 기능에 대한 이해를 높일 수 있었고 그 덕에 유사한 작업을 진행할 때 도움이 되지 않을까 싶다.

이번 프로젝트는 팀원들과의 소통이 전 프로젝트보다 잘 되어 큰 어려움 없이 협업을 통해 진행이 원만하게 흘러갔다고 생각한다.

확실히 프로젝트에서는 소통이 가장 중요하다는 것을 깨달았고 좋은 경험인 것 같다.

김동진

프로젝트를 통해 다크웹, 딥웹, 서피스웹에 대한 구분과 이해도를 높일 수 있었다. 알림시스템을 위해 Telegram, Slack API를 활용한 봇을 개발해보면서 봇을 활용한 알림시스템을 처음 만들어보았다. selenium, beautifulsoup4를 사용하여 크롤링 스크립트를 제작해보면서 웹크롤링에 대한 이해와 방법을 학습하게 되었다. 덕분에 웹에 대해서 더 자신감을 갖게된 것 같다. 유출에 대한 알림을 해주기 위해 지속적이고 짧은 주기의 크롤링 방법을 고민하다가 cron이라는 작업 스케줄러를 알게되고 사용할 수 있게되었다. 크롤링한 데이터를 보관하고 웹에 게시하는 것에 대한 고민하면서 json파일로 저장하고 데이터베이스에 저장함으로써 데이터베이스 부하를 줄이고 안전하게 웹에 게시하는 것까지 완벽했다고 생각한다. 아쉬운 점이 있다면 캡차우회를 시도해보고 싶었는데 선정된 웹사이트들이 캡차가 없었다는 것과 크롤링한 데이터를 좀 더 디테일하게 분류하고 분석하여 좀 더 높은 수준의 데이터를 제공하고 싶었는데 아직 역량과 시간이 부족했던 것 같다.

프로젝트를 진행하면서 각자 역할을 맡아 개발하면서 문제가 발생하면 같이 고민하고 해결해 나간 것이 좋았다고 생각한다. 덕분에 팀원들도 다 같이 이해도를 올리고 점점 프로젝트 개발에 속도가 붙었던 것 같다.

장다솜

다크웹에 대한 인식은 부정적이고, 접근 자체가 위험하다고 여겨지고 있지만 프로젝트를 통해서 구조와 동작 방식에 대해 배울 수 있는 좋은 기회였다.

중간에 아파서 참석을 못한 날도 있어서 개발에는 큰 참여가 어려웠지만 flow chart를 만들면서 흐름을 이해하기 좋았고 팀원분들과의 소통과 협력으로 인해 막막했던 프로젝트를 잘 끝마친 것 같다.

프로젝트들을 진행하면서 이 경험들은 개인의 역량뿐 아니라 팀워크가 얼마나 중요한지 깨닫게 되는 좋은 경험이었다고 생각한다.

조운지

이번 프로젝트는 다음 메인 프로젝트를 위해 팀 자체가 성장하는 시간이 되었으면 하는 바람이 있었는데, 협업과 분업화가 더 잘된 프로젝트였고 결과물도 잘 나와서 기쁘다.

개인적으로는 다크웹, 크롤링, 알람 API, AWS 등에 대한 이해도를 높일 수 있었고, google one이나 HIBP 같은 서비스 수준까지 진행해보고 싶었는데 역력이 되지 않아 아쉽다.

하지만! 가장 아쉬운 점은 좋은 팀원들과 마지막 프로젝트를 앞두고 있다는 점... 마지막은 언제나 슬픈 법인데, 이번만큼은 웃으며 마무리를 할 수 있었으면 좋겠고, 모두의 기억에 남을 만한 프로젝트가 되었으면 한다. 화이팅!

최승희

프로젝트를 통해 다크웹에 대한 이해도를 높일 수 있었다. 웹사이트 선정 과정에서 다크웹, 딥웹을 조사하면서 얼마나 정보 유출과 불법행위가 쉽게 이루어지는지 보면서 다소 충격을 받았다. 또한 이번엔 처음으로 크롤링을 해보았는데 이를 통해 자연스레 웹페이지 구조를 파악하는 능력이 향상된 것 같다. 웹에 대해서는 개인적으로 어렵게 느껴지는 분야였는데 이제는 너무 친근해졌다는 것에 만족한다.


그리고 이번 프로젝트를 진행하면서 프로젝트라는 것을 어떻게 진행해나가야 하는지를 알 수 있었다. 지난번 프로젝트에서 멘토님이 피드백을 주실 때 프로젝트를 임하는 마음에 대해 언급을 하신 적이 있다. 그에대해 이제는 확실히 이해를 할 수 있을 것 같다.

이번 프로젝트는 유독 팀원간의 협업과 의사소통이 잘 되었다고 느낀다. 서로 간의 대화가 가장 많이 이루어졌던 프로젝트였는데 그로인해 더 좋은 결과물이 나왔다고 생각한다. 항상 든든한 팀원들이 있음에 감사합니다.

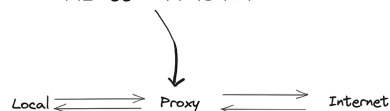
4. 참고문헌

[Python] Selenium Proxy를 이용한 Tor 사용하기

HTML 삽입 미리보기할 수 없는 소스 개요 6월에 유튜브를 보던 도중 Selenium으로 Proxy를 사용하던 하나의 영상을 보게 되었습니다. 원래 Selenium으로 Proxy를 사용하는 방법은 6월의 ToDoList 중 하나였는데 이제야 조사를 마쳐서 글을 쓰게 되었습니다. 아마 이 글의 제목을 보고 이 글을 읽으려고 하시는 분들은 Proxy를 적어도 한 번 들어

 <https://jakpentest.tistory.com/entry/Selenium-Proxy와-OSX에서-Selenium으로-Tor-사용하기>

이거를 Selenium에서 사용해보자



Flask 웹 서버 AWS EC2에 배포하기

Intro 지난 번 글에서 Flask 웹 프레임워크를 통해 간단한 딥러닝 웹 애플리케이션을 개발해보았다. 하지만 로컬(local) 환경에서 개발하였기 때문에 개발 서버를 종일 켜놓거나 고정 도메인을 따로 받지 않은 이상 외부 IP로 접근은 불가능하다. 그렇기 때문에 나쳐

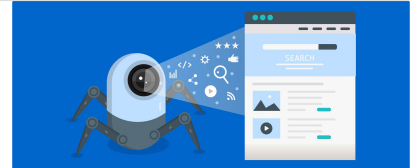
<https://velog.io/@jaehyeong/Flask-웹-서버-AWS-EC2에-배포하기>



비개발자가 쉽게 설명하는 웹 크롤링(Crawling)과 웹 스크래핑(Scraping)의 차이점

여러 플랫폼들은 '웹 크롤링(Web Crawling)'과 '웹 스크래핑(Scraping)' 기술을 활용하여 인터넷의 수많은 데이터 속에서 고객이 원하는 정보만을 골라 보여줍니다. 비슷한 듯 다른 크롤링과 스크래핑, 두 개념의 차이점은 무엇일까요?

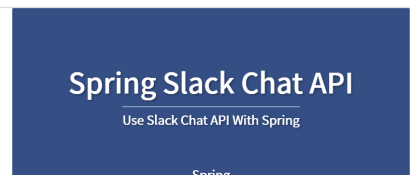
https://blog.hectodata.co.kr/crawling_vs_scraping/



Slack Chat API 사용하여 슬랙 알람 메시지 전송

슬랙에서 지원해 주는 Chat API를 사용하여 알람 메시지를 전송해 주는 샘플 코드를 작성해 보겠습니다. [STEP 1] 알람봇으로 사용할 App 생성 api.slack.com/apps 접속 후 Create an App 클릭 앱 이름 및 알람봇을 적용할 Workspace 지정 [STEP 2] 생성한 알람봇에게 권한 부여 좌측 사이드바 Features → OAuth & Permissions 클

<https://gcpower.tistory.com/entry/Slack-Chat-API-사용하여-슬랙-알람-메시지-전송>



Ransomfeed

An Italian project to track cyber gangs and store results in MySQL database to generate free RSS feeds

<https://ransomfeed.it/stats.php?page=group-stats-month>