[[1]](#footnote-1)

*Anti-Methods for Distributed Web-Crawler*

(Feburary 2018)

First A. Author, *Fellow, IEEE*, Second B. Author, and Third C. Author, Jr., Member, IEEE

*Abstract*—본 논문에서는 proxy를 사용해 IP를 분산시킨 distributed crawler에 대한 방어 기법으로서 node reducing을 제시하고 이에 대한 실험 방식과 결과를 소개한다.

*Index Terms*—Enter key words or phrases in alphabetical order, separated by commas. For a list of suggested keywords, send a blank e-mail to [keywords@ieee.org](mailto:keywords@ieee.org) or visit <http://www.ieee.org/organizations/pubs/ani_prod/keywrd98.txt>

# INTRODUCTION

웹 크롤링은

이 때 두 가지 문제가 존재한다. 하나는 웹 서비스 데이터 배포 방식이 업데이트를 거치면서 변경 될 수 있다는 점이며(입력 데이터의 구조 변경 사항을 반영하여 crawler를 업데이트 해야 한다), 나머지 하나는 일부 웹 서비스의 경우 anti-crawling 기법을 도입하여 crawling을 수행하는 측에서 crawler 개발에 많은 시간을 소요 하도록 한다는 점이다.

본 논문에서는 기존 안티 crawling 기법과 그에 대한 카운터 기법을 소개함으로써 기존 anti crawling 기법으로 분산형 crawler를 방어하지 못 하는 점을 설명한다. 또한 분산형 crawler라 사용하는 IP 셋을 점차적으로 black-list에 추가하는 새로운 안티 crawling 기법을 소개한다.

# Back Ground

### Http Archive

Http Archive는 Internet Explorer와 Chrome 등의 브라우저에 내장된 개발자 도구를 통해서 얻는 네트워크 로그를 저장한 파일 포맷이다. 네트워크 로그는 JSON 형태의 데이터를 가진 .har 확장자 파일로 저장된다.

HAR 파일은 로그 자체에 대한 메타 데이터와 request, response 내용에 대한 entries 데이터로 구성된다. 메타 데이터에는 브라우저의 종류와 버전, 생성 시간 등의 정보가 포함되며, entries 부분에는 시작 시간, 소요 시간, request와 response 데이터가 각 entry 별로 포함된다.

Entry 내의 request와 response 아이템에는 개발자 도구를 통해서 확인 가능한 모든 데이터가 포함되어 있다. 이는 개발자가 har 로그를 확인함으로써 실제 사이트 동작을 실시간 관찰한 것과 같은 양의 정보를 얻을 수 있다는 의미이며, 나아가서 har 로그를 parsing 함으로써 crawler 개발에 필요한 데이터를 자동 추출하고, crawling library 코드 작성까지도 반자동화 하는 것이 가능하다는 것을 의미한다.

### 멱법칙(Power Law)

멱법칙에 의하면 항목을 사용빈도 순서로 정렬 시, 순위가 하나 감소할 때 마다 사용빈도가 지수적으로 감소한다.

이는 web traffic에도 일반적으로 적용되며[1], 대부분의 web traffic은 사용빈도가 높은 일부 항목에 집중된다.

본 논문에서도 web traffic이 멱법칙을 따른다는 가정하에 long-tail 영역의 항목을 활용하여 crawler set을 식별하는 기법을 소개한다.

# Generating Crawling Library: HAR2LIB

본장에서는 Intelligent Crawler의 예시로서 har 로그 파일을 파싱하여 crawling library 코드를 생성하는 har2lib 패키지를 소개한다.

## Parsing HAR Files

HAR2LIB에서는 harlib 클래스를 제공한다. 클래스 객체를 생성하면 har 로그 파일을 로딩 한 다음 파싱을 진행한다. 파싱은 예외 처리, 헤더 분석, URL 분석 단계로 나눌 수 있다.

## Generating Crawling Library

파싱이 완료되면 HAR2LIB는 내부의 harlib.\_gen\_py() 메소드를 호출하여 해당 사이트에 대한 crawling method를 포함하는 python 클래스를 작성한다. 그 다음, HTTP 헤더를 브라우저에서 전송하는 것과 동일하게 설정 할 수 있도록 request 헤더 정보를 메소드 내부에 dict 형태로 저장한다.

생성된 클래스는 어디 까지나 crawling에 필요한 메소드를 포함할 뿐이며, 실제로 이를 호출하는 비즈니스 로직은 직접 구현해야 한다. 이는 crawling에 필요한 데이터를 코드화 시키는 부분과 crawling 시나리오를 구현하는 부분이 별개의 이슈이기 때문이다.

# Anti-Crawling Methods and Counter Scenario

Harlib가 crawling scenario를 직접 제시하지는 않지만 automated dealy와 같이 crawling 시나리오를 작성하는데 도움을 주는 기능은 있다. 본 장에서는 crawler를 감지하고 대응하기 위한 anti-crawling 기법과 이에 대한 우회 기법을 소개한다.

## Anti-Crawling Methods

웹 서비스에 대한 과도한 crawling을 하는 crawler를 감지하고, 자동으로 blocking을 수행하는 것은 서비스 운용 차원에서나 지적 자산 보호 차원에서 매우 중요하다. 본 절에서는 기존에 알려진 Anti-Crawling 기법들을 소개한다.

### *HTTP Header Check*

일반적인 crawler의 경우, 브라우저에서 사용하는 http 헤더를 사용하지 않는다. 서버에서는 클라이언트에서 오는 request 헤더를 확인하여 User-Agent 등의 값이 정상적으로 포함되어 있는지 확인하는 방식으로 정상 사용자와 crawler를 구분한다.

### *Access Frequency Recognition*

공격적인 crawling을 수행하는 공격자는 해당 웹서비스에서 수집하고자 하는 핵심 데이터를 사전에 정의하고, 불필요한 리소스를 요청하지 않고 특정 데이터를 요청하는 crawler를 구현한다. 이 경우, 웹서버는 특정 자원에 대한 호출만을 연속적으로 대량 처리해야 한다. 예를 들어 대한민국 소재의 560만개 지번에 대한 부동산 전수 데이터를 보유한 웹서비스의 데이터를 모두 복제 하려는 공격자라고 가정하자. 해당 웹서비스는 지번을 검색하면 html, js, css 파일을 클라이언트에 넘기고, html table form을 채워주는 데이터를 ajax 호출로 동적 완성한다. 공격자는 ajax 호출만을 python 등의 스크립트로 구현하고, 공개된 지번 주소 목록을 통해 확보하여 병렬적으로 서버에 데이터를 요청할 수 있다. 주의 깊은 공격자라면 서버의 쳐리 성능 한계에 근접하지 않도록 하겠지만 그렇지 않다면 자신이 원하는 데이터를 최대한 빠른 시간 내에 얻을 수 있도록 수집 프로세스를 병렬화 할 것이다.

### *Access Pattern Recognition*

접근 패턴 인식은 상기 접근 빈도 인식과 더불어 동적 분석을 통한 Anti-Crawling 기법이다. 클라이언트가 정상적으로 요청해야 하는 리소스에 대한 호출 없이 특정 리소스만을 연속적으로 요청 한다면 해당 IP를 block한다. 접근 패턴 인식 방식을 고도화 하여 적용한 웹서비스의 경우는 사용자 UX 관점에서 서비스를 일련의 연속적인 요청들의 집합으로 보고, 같은 집합 내에 속하는 요청과 응답을 Cookie에 특정 Hash 값을 포함시키는 방식으로 chaining하여 이 중 일부 요청을 별도로 할 수 없도록 구성한다.

## Anti-Cralwer Counter Method

앞서 3장에서 설명한 har2lib를 통해서 생성된 지능형 크로울러 라이브러리를 활용할 경우, 앞서 설명한 3가지 anti-crawling 방식에 다음과 같이 모두 대응 가능하다. 1~3번 기법을 모두 사용하는 distributed crawler는 앞서 설명한 anti-crawling 기법만으로는 방어하기 어렵다.

### Request Header Replay

har2lib에 의해서 생성된 모든 요청 method는 정상적인 브라우저에 의해 요청한 것과 동일한 http request header를 자동으로 탑재한다. 웹서비스는 http 헤더 확인으로 크로울러와 브라우저를 구분할 수 없다.

### *Access Frequency Auto Configuration*

har2lib는 har 파싱 과정에서 각 method 별로 실제 소요된 시간 데이터를 사용하여 method 별로 automated dealy time value를 설정한다. 모든 method는 auto dealy 옵션을 True로 줄 경우 실제 소요된 것과 같은 delay를 sleep 함수를 통해 manipulate한다. 설정을 적용한 후 다중 IP 프록시 서버와 결합하면 IP 당 접근빈도 한계를 넘기지 않는 유닛으로 구성된 병렬 crawling 네트워크를 구성 가능하다.

### *Access Pattern Replay*

har2lib는 비록 크로울링 시나리오를 직접 작성해주지는 않지만, ajax 호출 복제만으로 구성된 시나리오가 접근패턴인을 우회하는데 도움이 되는 접근패턴에 대해 가이드를 지원한다. har 로그 자체가 정상적인 접근 패턴을 기록한 것이기 때문에 이를 바탕으로 접근 패턴을 별도의 도표로 요약 가능하다.

# Blocking Distributed Crawler

본 장에서는 기존의 anti-crawling 기법으로 방어 할 수 없었던 distributed crawler를 detect하고 block하기 위한 새로운 기법을 제시한다.

## Number of Distributed Nodes

Distributed crawler가 웹사이트의 전수 데이터를 복제하기 위해서는 다음 조건을 만족시켜야 한다. 대상 사이트에서 월 단위로 업데이트 되는 아이템의 개수(Um), 해당 사이트에서 제한하는 일당 최대 접속 회수(Td), crawling을 수행하는 IP의 개수(Cn)에 대해서 Cn Um / (Td \* 30)을 만족해야 한다.

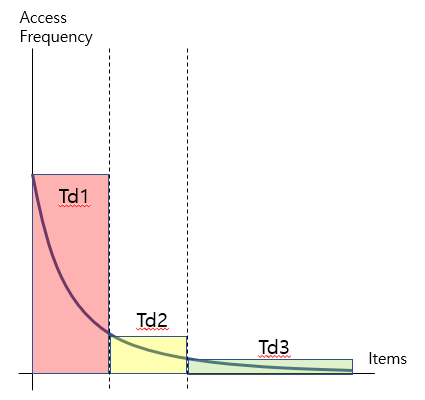
예를 들어 월간 60,000개의 데이터를 업데이트하는 서비스에서 하루 최대 접속 회수를 50회로 제한할 경우, 공격자는 최소 40개의 분산된 IP를 사용하여 crawling을 수행해야 한다. 반대로 서비스 제공자 측에서는 Um이 클수록, Td가 작을수록 유리하다. 그러나 Um은 임의로 확보하기 어려우며, Td를 줄일 경우 정상적인 사용자에 대해 false positive를 발생시키는 비율이 증가한다.

## Node Reducing with statistical approach

서비스 제공자 측에서 Um을 증가시키거나 Td를 줄이는 대신 Cn 중 일부를 crawler로 식별하고 block 함으로써 Cn 을 감소시키는 방법이 있다.

공격자가 사용하는 IP 중에서 웹사이트 서버 측이 crawler인 것을 확인한 IP의 개수를 Cm이라고 할 때, 공격자는 Cn - Cm Um / (Td \* 30)을 만족시켜야 한다. 서비스 제공자 측에서는 Cm Cn - Um / (Td \* 30)를 만족하는 Cm을 구하면 된다.

서비스 제공자는 통계적인 기법을 사용하여 Td를 일괄적으로 축소하지 않고도 block-list를 작성 가능하다. 아이템 별로 사용자들이 접속하는 빈도는 동일하지 않으며, 가장 접속 빈도가 높은 항목이 왼쪽에 오도록 정렬하면 멱법칙에 따라서 다음과 같이 지수적으로 감소하는 그래프 형태로 분포한다.



공격자가 서비스로부터 전수 데이터를 복제하기 위해서는 long-tail(Td3) 구간에 있는 아이템에 대해서도 접근을 해야 한다. 그러나 공격자는 자신이 접근하는 아이템이 어느 구간에 속하는 아이템인지 정확히 알지 못 한다. 서비스 제공자는 이와 같은 정보의 비대칭성을 사용하여 long-tail 구간에 일반 사용자들 보다 자주 접근하는 IP들을 쉽게 식별 가능하다. Long-tail 구간을 통해서 Cm 값을 증가시키기 시작하면 공격자는 보다 적은 수의 IP로 crawling을 하게 되고 Td2 구간에서도 Cm이 증가하게 된다.

구간 별 Td 값은 해당 구간 아이템 중에서 IP 당 일 접근 빈도가 가장 높은 아이템의 접근 빈도 값(Amax)에 해당 구간 접근 빈도의 표준편차(s)를 다음과 같이 더한 값으로 한다.

Td = Amax + s \* 2

특정 IP가 상기 공식에 의해 결정된 Td값 이상의 회수로 long-tail 영역에 있는 아이템에 접근한다면 block-list에 포함시킬 수 있다.

## Dummy Items

서비스 제공자는 실제 서비스 대상인 아이템 외에 crawler를 검출하기 위한 dummy item을 포함시킬 수 있다. 해당 아이템은 일반 사용자가 UI를 통해서 정상적으로 접근 불가능하다. 예를 들어 HTML 태그로 존재하지만 attribute 설정 등으로 화면에는 표시되지 않도록 하거나, index 상에는 존재하지만 현실세계에 없는 값이어서 일반 사용자가 관심을 갖지 않는 경우를 말한다.

이와 같은 dummy item은 순차적으로 접근을 수행하는 crawler는 접근하지만 일반 사용자는 접근 가능성이 낮기 때문에 상대적으로 매우 낮은 threshold 값을 유지 할 수 있으며, traffic을 통해 도출한 long-tail 구간 보다 더 낮은 구간을 생성 할 수 있다.

# Experiment

상기 내용을 검증하기 위해서 실제 웹트래픽 데이터에 대해서 크롤러 IP를 분류하는 실험을 수행하였다. NASA에서 공개한 1달치 분량의 웹 트래픽을 기반으로 진행 하였으며 데이터에 대한 상세 내용 및 실험 방식은 다음과 같다.

## Web Traffic Data

### Source

NASA에서는 1995년 7월 한달 간 발생한 총 1,891,715건의 접속 로그를 공개하였다. 본 논문에서는 해당 로그를 csv 형태로 파싱하여 IP, date, access target 그리고 access result 까지 총 4개 column으로 구성하였다.

접속한 총 IP는 81,978개이며 아이템은 21,649개이다. 가장 많이 접속된 아이템은 ‘/images/NASA-logosmall.gif’로서 111,116회 요청을 받았다.

### Traffic Distribution

전체 로그에 대해서 access target 별로 접속 회수를 연산하여 가장 접속이 많은 순서대로 sorting 하였으며, 그 결과는 5장에서 언급한 바와 같이 멱법칙이 적용되는 형태로 분포하는 것을 확인하였다.

또한 하기 Figure 2, 3, 4와 같이 Td1, Td2, Long-tail 각 구간 내부적으로도 power distribution형태인 것을 확인하였다. Figrue 2는 Td1에 해당하는 상위 0.5%에 해당하는 38개 item의 접속빈도 그래프이며, Figure 3는 Td2에 해당하는 상위 100~2000개, Figure 4는 상위 0.5~1%에 해당하는 100~2000 이하에 대한 그래프이다. 다음 절의 시뮬레이터에서는 앞서 설계한 내용대로 Long-tail에 속하는 아이템 집합을 사용하여 node reducing을 수행할 것이다.



Figure . Access Count in Td1



Figure . Access Count in Td2



Figure . Access Count in Long-tail

## Simulation

본 논문에서는 2가지 목적의 시뮬레이션을 구현하였다. 하나는 Long-tail에 속하는 아이템들을 통해 node reducing을 수행하여 크롤러 IP 집단을 검출 및 무력화하는 것이 가능한지 여부를 확인하는 것이고, 나머지 하나는 동일한 검출 로직에 실제 트래픽을 입력하였을 때 false positive를 확인하는 것이다.

### Data Pre-Processing

시계열 데이터에서 모델링에 사용된 데이터와 실험 데이터가 중복되는 것을 방지하기 위해서 Long-tail는 30일치 데이터 중에서 1일 부터 24일 까지의 데이터를 사용하여 구성하였으며, 트래픽 검증은25일 부터 마지막 날 까지의 데이터를 사용하여 수행하였다.

Html 파일 접근 시, gif 확장자 파일들에 함께 접근하면서 누적 접근 수치가 중복으로 증가하는 것을 방지하기 위해 gif 확장자 파일들은 long-tail 그룹에서 제외하였다.

마지막으로 접근에 성공하지 못 한 request 로그는 실험에서 제외하였다.

### Simulators

시뮬레이터는 python을 사용하여 구현하였으며, 인자로는 크롤러가 사용하는 분산 IP집합의 크기와 Long-tail 목록, 전체 아이템 리스트, 그리고 검출에 사용하는 threshold 값을 입력 받도록 하였다.

구현 방식은 Crawler IP Set이 전체 아이템 리스트를 순회하며 각 아이템에 접근하도록 하였으며, 접근 시 마다 크롤러 분산 IP집합 내에서 IP를 변경하며 접근하도록 하였다.

크롤러가 Long-tail항목에 접근할 경우, 해당 IP를 warning dictionary에 추가하고 access count를 1 증가시킨다. 단, 같은 IP가 동일한 항목에 접근하는 경우에는 crawling 목적과는 관계가 없기 때문에 access count를 증가시키지 않는다. Access count가 threshold 값을 초과하면 해당 IP를 banned list에 추가하는 방식으로 Node Reducing을 구현하였다. 다음 Figure 5는 7,649개의 아이템에 대해서 100개의 분산 IP를 사용하는 크롤러를 동작시킨 예시이다. Long-tail 아이템의 개수는 5,355개이며 임계치는 20으로 설정하였다.



크롤러 IP set에 포함된 IP들은 점차적으로 access count를 누적하게 되고, 크롤러 노드 집단 전체의 access count가 {number of nodes} \* {threshold} 이상으로 증가하는 시점부터 node reducing이 시작된다.

시뮬레이터의 다른 기능은 실제 웹트래픽 로그를 바탕으로 request 요청을 Crawler simulator에 입력하는 기능이다. 이는 시뮬레이터가 실제 웹트래픽을 crawler로 판단하는 케이스를 확인하기 위한 목적으로 구현하였다.

### Node Reducing Result

실험은 threshold를 30으로 설정한 상태에서 진행하였으며, 최대 222개의 노드로 구성된 crawler set을 완전히 검출 가능했다. 노드의 개수가 300개가 넘어갈 경우 전혀 검출되지 않았다. False positive는 일 평균 1.33건으로, 일 평균 접속 IP의 개수 3,631개 대비 0.0312%로 확인하였다. 다음 Figure는 222개 노드로 구성된 crawler set이 simulator 상에서 감소되는 과정을 그래프로 나타난 것이다.



이는 1995년 NASA traffic data를 기반으로 한 결과이며, 사이트가 보유한 아이템의 개수나 long-tail의 길이에 따라서 더 많거나 적은 crawler set에 대해 node reducing을 적용 가능하다.

False positive는 6일치 데이터에서 총 8건 발생하였으며, 중복 검출을 제외하면 6개 IP가 crawler 노드로 인식되었다. 각 IP별로 한 달간 발생시킨 request 건수는 다음과 같다.

|  |
| --- |
| 156.80.168.122: 117  163.205.180.17: 564  dwkm206.usa1.com: 167  jalisco.engr.ucdavis.edu: 424  jbiagioni.npt.nuwc.navy.mil: 2124  sputnix.cas.und.nodak.edu: 101 |

상대적으로 적은 request를 발생시킨 156.80.168.122와 sputnix.cas.und.nodak.edu가 crawler node로 검출된 이유는 해당 IP들의 요청이 특정 날짜에 집중되었으며, 이 중 29.7% 가 long-tail 영역의 아이템을 대상으로 발생하였기 때문이다.

# Conclusion

본 논문에서는 web traffic이 멱법칙을 따르는 성질을 이용하여 distributed crawler의 IP set을 식별하고 점차적으로 IP를 감소시키는 node reducing 기법을 소개하였다.

Node reducing 기법은 다수의 IP를 사용하는 distributed crawler를 상대로 매우 낮은 수준의 false positive를 유지하며 효과적으로 crawler set을 식별하는 결과를 보여주었다.

# Future Works

Web traffic은 일반적으로 특정 시점에 traffic burst가 발생하는 경향이 있다.[1] 본 논문의 실험은 실제 발생한 traffic log에 기반하였음에도 불구하고 실험에 사용한 데이터의 시점이 한달간이기 때문에 새로운 아이템이 추가되거나 이슈가 발생하여 traffic burst가 발생하는 케이스를 포함하지 않았다.

본 논문의 결과를 실제 서비스에 더 안전하게 적용하기 위해서는 Traffic burst 발생 케이스에 대한 실제 traffic 데이터를 기반으로 long-tail 영역의 항목 이동 수준과 threshold 값을 유지할 수 있는지 여부 등을 확인하는 연구가 필요하다.

References

1. M.V Simkin and V.P. Roychowdhury, “A theory of web traffic” https://arxiv.org/pdf/0711.1235.pdf

1. This paragraph of the first footnote will contain the date on which you submitted your paper for review. It will also contain support information, including sponsor and financial support acknowledgment. For example, “This work was supported in part by the U.S. Depart­ment of Com­merce under Grant BS123456”.

   The next few paragraphs should contain the authors’ current affiliations, including current address and e-mail. For example, F. A. Author is with the National Institute of Standards and Technology, Boulder, CO 80305 USA (e-mail: author@ boulder.nist.gov).

   S. B. Author, Jr., was with Rice University, Houston, TX 77005 USA. He is now with the Department of Physics, Colorado State University, Fort Collins, CO 80523 USA (e-mail: author@lamar.colostate.edu).

   T. C. Author is with the Electrical Engineering Department, University of Colorado, Boulder, CO 80309 USA, on leave from the National Research Institute for Metals, Tsukuba, Japan (e-mail: author@nrim.go.jp). [↑](#footnote-ref-1)