Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.30, NO.3, Jun. 2020 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2020.30.3.481

사용자 경험을 고려한 규칙기반 악성 URL 탐지 라이브러리 개발*

김 보 민,[1†] 한 예 원[1], 김 가 영[1], 김 예 분[1], 김 형 종[2‡]
[1,2]서울여자대학교(학생, 교수)

Development of Rule-Based Malicious URL Detection Library Considering User Experiences*

Bo-Min Kim,[1†] Ye-Won Han[1], Ga-Young Kim[1], Ye-Bun Kim[1], Hyung-Jong Kim[2‡]
[1,2] Seoul Women's University(Undergraduate, Professor)

요 약

악성 URL의 전송을 통한 악성코드 전파 및 불법적 정보 수집은 정보보안 분야의 가장 큰 위협 중의 하나이다. 특히, 최근 스마트폰의 보급으로 인하여 사용자들이 악성 URL에 노출될 확률이 더욱 높아지고 있다. 또한, 악성 URL을 노출 시키는 방법 역시 다양해 지고 있어서 이를 탐지하는 것이 점점 어려워지고 있다. 본 논문은 악성 URL에 대한 사용자의 경험에 대한 설문을 진행한 후, 이를 고려하여 악성 URL을 규칙기반으로 탐지하기 위한 라이브러리 개발 연구를 다루고 있다. 특히, 본 연구에서는 독자적인 규칙을 기반으로 악성 URL을 탐지 하기 위해 Rule-set을 정의하고, Rule-chain을 생성하여 악성 URL 탐지의 확장성을 제시하고 있다. 또한 어떤 애플리케이션에서도 활용이 가능한 라이브러리 형태로의 개발을 통해 다양한 응용프로그램에서 활용할 수 있도록 하였다.

ABSTRACT

The malicious URLs which can be used for sending malicious codes and illegally acquiring private information is one of the biggest threat of information security field. Particularly, recent prevalence of smart-phone increases the possibility of the user's exposing to malicious URLs. Since the way of hiding the URL from the user is getting more sophisticated, it is getting harder to detect it. In this paper, after conducting a survey of the user experiences related to malicious URLs, we are proposing the rule-based malicious URL detection method. In addition, we have developed java library which can be applied to any other applications which need to handle the malicious URL. Each class of the library is implementation of a rule for detecting a characteristics of a malicious URL and the library itself is the set of rule which can have the chain of rule for deteciing more complicated situation and enhancing the accuracy. This kinds of rule based approach can enhance the extensibility considering the diversity of malicious URLs.

Keywords: Mailcious URL Detection, Rule Base, Rule-chain, Android Programming, Java Library

I. 서 론

공격자들은 특정 웹페이지를 조작하여, 해당 페이지의 URL을 유포하고, 페이지에 접속하는 일반 사용자의 장치에 악성코드가 자동으로 다운로드 되도록 한다[1]. 최근에는 스마트폰 사용자가 늘어나면서

Received(04. 21. 2020), Modified(05. 22. 2020), Accepted(05. 29. 2020)
* 이 연구는 2017년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구(NRF-2017R1D1A1B030 34644)이며, 본 논문은 2020학년도 서울여자대학교 교내 연구비의 지원을 받았음(2020-0313).
† 주저자, kbm17@naver.com
‡ 교신저자,hkim@swu.ac.kr(Corresponding author)

MMS와 인터넷 접속을 통한 스미싱(Smishing) 공격이 늘어나고 있다. 사용자의 호기심을 자극하는 문자메시지 내용에 인터넷 URL을 첨부하여, URL 클릭을 유도한다. 사용자는 URL을 클릭함과 동시에 개인정보 유출 혹은 소액결제 등의 피해를 입을 수 있다. 스미싱을 이용한 공격들은 앞으로 더 다양해지고 교묘해질 것으로 예상된다[2]. 현재 스마트폰의 등장과 함께 SNS를 통한 사용자 간 메시지 교류가 폭발적으로 늘고 있는 추세이다[3]. 따라서, 공격자 혹은 악성 URL임을 인지하지 못한 피해자가 SNS로 URL을 공유한다면, 이로 인한 피해의 규모가 급격히 커질 수 있다.

본 논문에서는 규칙기반 지식 표현 체계를 가지고 사이트의 악성 확률을 반환해주는 'HypoChain1.1.2' 라이브러리 개발에 대한 연구 과정 및 결과를 제시한다. 라이브러리 개발을 통해 자바 기반의 다양한 메신저 플랫폼에 쉽게 이식할 수 있도록 확장성 및 호환성을 제공하고자 한다.

2장은 기존의 연구에 대해 제시한다. 3장에선 사용자의 행위적 특성을 고려한 연구의 필요성에 대해 제시한다. 4장에서는 악성 URL 탐지 규칙에 대해 제안하며, 탐지 기준이 되는 Rule-set에 대한 설명과 Rule-set을 기반으로 한 Rule-chain의 생성 및 의미에 대해 다룬다. 5장에서는 규칙 및 Rule-chain을 바탕으로 한 HypoChain1.1.2 라이브러리의 전체 구성 및 동작 원리에 대해 제시한다. 6장에서는 본 연구를 통한 탐지규칙 추가 및 삭제의 용이성 및 동적인 Rule-chain 구성에 대해 논의하며, 7장에서는 Android 환경적용 결과에 대해 제시한다. 마지막 8장에서는 결론 및 향후 연구 계획에 대해 제안한다.

II. 기존 연구

악성 URL을 통한 악성코드 감염 및 개인정보 침해를 방지하기 위한 많은 기존 연구들이 존재한다. 한국인터넷 진흥원에서는 2005년부터 악성코드 은닉 웹사이트를 탐지하기 위한 'MCFinder'을 사용하고 있으며, 이는 악성 사이트의 은닉 기법 및 경유지, 유포지 정보 등을 패턴으로 등록하여, 패턴과의 일치 여부를 비교하고 탐지하는 '패턴 매칭' 방식으로 구현되어있다[13]. WebKit API를 활용한 연구도 존재한다. WebKit API의 일부를 수정하여 탐지 URL을 미리 설계된 악성 URL 데이터베이스와 비교한 뒤, 존재하지 않는 경우 URL을 크롤링하여 탐지하도록 한다[14]. 기존의 모바일에서 탐지용으로 많이 쓰이는 악성 사이트 차단 앱 중에선 안랩의 v3 mobile plus와 알약 m이 대표적이다. URL이 포함된 문자메시지의 악성 앱 또는 악성코드 탐지 및 메시지 차단이 대표적인 기능이다[15][16].

'MCFinder'는 최신 악성 사이트마다 특징을 모두 추출하여 '패턴화'해야 하므로, 소스코드를 처음부터 끝까지 구현해야 한다[13]. WebKit API를 활용한 연구는 모바일 전용 악성 링크에 대해서 아직 판별할 수 없으며[14], 기존의 모바일에서 탐지용으로 쓰이는 모바일 악성 사이트 차단 앱은 사용자가 앱을 설치하지 않으면 탐지 자체가 불가능하다는 한계점이 있다. 이를 통해, 소스코드를 최대한 간결하게 구현하며, 모바일뿐만 아니라 어느 소프트웨어에서든 적용이 가능한 악성 URL 탐지 라이브러리 개발에 대한 필요성을 느껴 본 연구를 진행하였다. 악성 URL의 특징을 추출하여 이론적으로 정리해놓은 선제 연구는 URL 주소 자체 및 HTML 소스코드를 기반으로 특징을 나누어 정리하고 있다[5][6]. HypoChain1.1.2 라이브러리는 선제 연구의 이론을 활용하여, 악성 탐지규칙 구현 및 Rule-chain 체계를 완성하였다. 이는 기존 탐지 방법에 비해 코드 구현이 비교적 간단하여 최신 악성 사이트의 특징을 바로 반영할 수 있다는 효율성을 제시한다.

III. 사용자의 행위적 특성을 고려한 연구의 필요성 제시

본 연구가 시작되기 전, 연구의 필요성을 제시하기 위해 간단한 객관식 문항 5개로 이루어진 설문조사를 실시한 바가 있다. 총 167명의 설문을 받았으며, 10대(7.8%), 20대(80.8%), 30대(1.8%), 40대(4.2%), 50대(5.4%)의 응답을 받았다. 그중 SNS의 이용이 가장 활발한 20대의 응답이 주를 이루었다.

'공유된 URL에 대한 당신의 행동 특성'이라는 제목의 설문이 진행되었으며, '카카오톡, 페이스북 메시지, 문자 등을 통해 상대방이 공유한 URL(링크)를 받아본 경험이 있다면, 평소에 어떻게 행동하는지?', '악성 URL 판단을 위한 메신저상의 대화를 읽어 들일 수 있는 권한을 활용할 것인지?'에 대해 물었다. 각 질문은 송신자를 잘 아는 경우(Well-known person), 송신자를 잘 알지 못하는 경우

(Unknown person)에 대해 각각 응답을 받았으며, 두 가지 측면에서 연구의 유효성을 제시해볼 수 있다.

3.1 라이브러리 활용에 대한 연구 유효성 제시

송신자로부터 URL을 받았을 경우 '의심 없이 바로 접속한다.', '절대 접속하지 않는다.' '백신 어플을 깔아 미리 확인 후 접속할지를 결정한다.' '기타 의견'으로 선택지가 구성되었다.

송신자를 잘 아는 경우, 대체로 79.6%의 조사자가 '의심 없이 바로 접속한다'에 응답했고, 10.8%는 '접속하지 않는다', 2.4%는 '백신 어플을 설치해 미리 확인 후 결정한다'라고 응답하였다.

송신자를 모르는 경우, Fig. 1.에서 보이는 바와 같이 대부분 85%가 접속하지 않는다고 응답하였고, 6%는 의심 없이 접속하는 편이라고 대답하였다. 백신 어플로 확인 후 결정한다에 응답한 조사자는 2.4%였다. 그 밖의 기타를 선택한 경우 '카카오톡 상에서 링크에 해당되는 미리 보기 이미지를 보고 들어갈지 말지를 결정해본다.' 등의 의견이 있었다.

통계를 보면 송신자를 잘 알거나 혹은 모르는 경우, 각각 79.6%, 6%의 응답자가 URL을 의심 없이 들어가 본다고 하였다. 만약 공유된 URL이 사용자의 정보를 탈취할 수 있는 스미싱 URL이거나, 공격자의 홈페이지로의 접속이 의도된 URL이라면, 의심없이 들어가 본다에 답한 응답자들은 피해를 입게 된다. 이를 제한하기 위해 실제로 백신 어플의 활용을 권장하지만, 실제로 167명 중 백신 어플을 따로 사용하여 링크에 접속하는 응답자는 전체의 2.4%에 그쳤다.

위와 같은 결과를 통해 모바일 환경에서의 URL 탐지는 중요하지만, 백신 어플의 실효성이 떨어진다는 것을 파악할 수 있다. SNS 플랫폼이나 메신저 앱을 개발하려는 개발자들은 HypoChain 라이브러리를 통해 탐지 기능을 어플 자체에 탑재하여 활용할 수 있다. HypoChain의 간결한 코드체계를 통해 외부에 간단한 메소드 호출로 독자적 악성 URL 탐지가 가능하다.

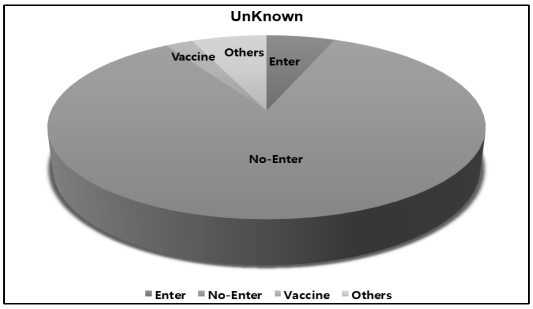

Fig. 1. URL from Unknown person

3.2 메시지 접근 권한 부여에 대한 고찰

HypoChain 라이브러리를 사용하는 App이 메시지에 대한 접근 권한을 가질 때 이를 활용하여 곧바로 악성 URL 여부를 확인하고 사용자에게 경고 알람을 주는 것이 가능하다. 본 설문에서 메시지 접근 권한 부여 여부에 대한 선택지는 'App이 메시지를 읽을 수 있는 권한을 허용한다.', 'App이 메시지를 읽을 수 있는 권한을 허용하지 않는다.'라는 문항으로 구성되었다. 설문 결과 지인의 메시지에 대해서는 61.1%가 권한을 허용하고자 하였고, 지인이 아닌 경우 66.5%로 권한을 허용한다고 하였다. 이러한 설문을 기반으로 만일 HypoChain 라이브러리의 탐지 기준을 통해 악성 URL을 100% 탐지할 수 있다면, 메시지에 대한 읽기 권한을 허용한 각각 61.1%, 66.5%의 이용자에게 정확한 자동탐지 능력을 제공할 수 있다고 예측할 수 있다.

IV. 악성 URL 탐지 규칙

4장에서는 HypoChain1.1.2에 활용된 규칙들에 대해 살펴본다. 다음은 탐지 규칙으로 활용되는 Rule-set과 Rule-chain에 대한 내용이다.

4.1 Rule-set

HypoChain은 기존에 이론적으로 알려져있는 악성 URL의 URL-based 및 HTML-based 특징을 구현하여 활용한다. 각각의 Rule은 클래스로 구현되어있으며, Rule을 모아둔 일종의 데이터베이스를 Rule-set이라 지칭한다.

4.1.1 URL-based Rule

URL-based Rule은 사용자로부터 입력받은 URL 주소 자체만을 보고 판단하는 기준이다.

Table 1.은 HypoChain에 클래스로 구성되어

Table 1. HypoChain URL-based Rule

RuleName	Description
AtMark	Whether url has “@” .
BackSlash	Whether url has “\”.
Bitly	Whether url shortened “Bitly” form.
ContainDigits	Whether url contains digits.
Dot	Whether url has “.”
EqualSign	Whether url has “=”.
Hypen	Whether url has “-”.
LengthOfURL	Whether length of url is over 65 words.
SemiColon	Whether url has “;”.
Tld	Whether url comes from nation where a lots of malicious urls are made.

있는 10개의 URL-based Rule에 대한 명칭 및 설명을 제시한다. AtMark, Hypen, SemiColon, EqualSign Rule 클래스는 URL 주소 자체에 ‘@’, ‘-’, ‘;’, ‘=’의 특수문자가 들어간 경우를 포착한다[5][6][7]. BackSlash, Dot Rule 클래스는 ‘\’, ‘.’ 문자를 포함하고 있는 경우를 의미하며, URL 주소를 통해 상위권한탈취 위험을 판단하는 기준에 사용한다[5][6]. Dot Rule 클래스는 URL에 IP주소가 포함되어 있는 경우를 판단할 때에도 재사용된다[5]. Bitly 클래스의 경우, Bitly API를 사용하여 생성된 단축 URL에 대한 기준을 의미한다. 악성 사이트의 URL을 단축화 하여 사용자의 의심 없는 접속을 유도할 수 있다[5][8]. ContainDigits 클래스의 경우 URL 자체에 접속 타겟 IP 주소가 들어있는 경우를 파악하기 위해 숫자 포함 여부를 판단하는 클래스이다[5]. LengthOfURL 클래스란 URL 주소 자체의 길이가 65를 넘는 경우를 의미한다[9]. 의도적으로 URL 길이를 길게 하여 사용자에게 URL 주소 내의 의심스러운 내용을 들키지 않게 하므로 악성 판단 기준으로 삼았다[5][6]. TLD 클래스란 TLD(Top Level Domain)를 보고 해당 URL이 어느 곳으로부터 왔는지를 파악할 수 있는 기준이다[6]. 해외 사이트의 여부를 탐지하는 역할로 현재 HypoChain에는 중국(.cn) 과 러시아(.ru)로부터 온 URL을 탐지할 수 있는 기준이 구현되어 있다[10][11]. 아래의 Fig. 2.는 AtMark Rule을 구현한 코드이다.

```
public class AtMark extends RuleBase {

    private String url;
    private boolean hypothesis = false;

    public boolean isHypothesis() {
        return hypothesis;
    }

    public void setHypothesis(boolean hypothesis) {
        this.hypothesis = hypothesis;
    }

    public AtMark(String url) {
        super(url);
        this.url = url;
    }

    public void rule(){
        char [] c ;
        c = url.toCharArray();

        for(int i=0; i<c.length; i++){
            if(c[i]=='@'){
                setHypothesis(true);
            }
        }

    }
}
```

Fig. 2. ‘AtMark’ Rule Class Definition

4.1.2 HTML-based Rule

HTML-based Rule은 사이트의 HTML 코드 정보를 보고 판단하는 기준이다.

Table 2.는 HypoChain에 클래스로 구성되어 있는 13개의 HTML-based Rule에 대한 명칭 및 설명을 제시한다. ATag 클래스는 〈a〉태그의 존재 유무를 확인하는 Rule이다. 〈a〉태그는 보통 하이퍼링크를 걸어주는 태그로 유용하게 쓰이나, 악성 사이트로 사용자를 링크할 수도 있는 위험요소가 존재한다[5]. HiddenTag 클래스는 hidden태그의 존재 유무를 확인하는 Rule이며, hidden 태그는 주로 사용자에게는 보이지 않지만 정보를 전달할 때 쓰이는 은닉형 태그이다. hidden태그를 통해 공격자가 사용자 몰래 악성 파일을 전송할 수 있다는 취약점이 존재한다[6]. IframeTag 클래스는 〈iframe〉 태그의 존재 유무를 판단하는 클래스이다. 〈a〉 태그와 같이 링크 태그와 함께 쓰였을 때, 악성 사이트의 특징을 나타낼 수 있다[5]. LengthOfDoc, NumberOfLines 클래스는 빠르게 만들어진 악성 사이트는 사이트 내의 정보가 부실하기에 이를 특징을 판단하기 위해 사용될 수 있다[6]. UnEqualHtmlTag 클래스 또한, 공격자가 빠르게 악성 홈페이지를 제작했을 시, 〈/html〉 태그로 마무리하지 않아 태그의 짝이 맞지 않는 경우를 의미한다[6], SuspiciousWords 클래스는 사이트 내의 의심스러운 단어를 탐지한다. HypoChain에는 택

Table 2. HypoChain HTML-based Rule

RuleName	Description
ATag	Wheter html code has 〈a〉 tags.
CharCodeAt Method	Whether html code has "charCodeAt()" method.
Escape Method	Whether html code has "escape()" method.
FromCharCodeMethod	Whether html code has "fromCharCode()" method.
HiddenTag	Whether html code has "hidden" input type.
IframeTag	Whether html code has 〈iframe〉 tags.
LengthOfDoc	Length of html document is too short.
NumberOfLines	Document lines is small in number.
ParseIntMethod	Whether html code has "parseInt()" method.
Suspicious Words	Whether html document has suspicious words.
UnEqualHtml Tag	Whether html code has unequal pair of 〈html〉-〈/html〉.
WindowOpen Method	Whether html code has "window.open()" method.

배 주소지 변경을 가장한 피싱 사이트의 특징인 '택배' 단어가 기준에 구현되어 있다[12].

그 외 CharCodeAtMethod, EscapeMethod, FromCharCodeMethod, ParseIntMethod, WindowOpenMethod 클래스는 악용될 수 있는 Java 함수가 포함되어 있는지 파악한다 [5][6]. 아래의 Fig. 3.은 ATag Rule의 구현 코드 일부인 오버라이딩 된 rule()메소드이다.

```
public void rule(){

    String[] checkBig = html.split("<");
    for(int i=0; i<checkBig.length;i++){
        String [] checkMid = checkBig[i].split(">");
        for(int j=0; j<checkMid.length; j++){
            String [] checkSmall = checkMid[j].split(" ");
            for(int k=0; k<checkSmall.length; k++){


                if(checkSmall[k].equals("a")){
                    setHypothesis(true);
                }
            }
        }
    }
}
```

Fig. 3. Overrided 'rule()' method of 'ATag' Rule

4.2 Rule-chain

Fig. 4.는 Rule-chain의 동작 원리를 나타낸다. Rule-chain은 4.1에 기술된 Rule-set의 Rule 클래스들을 가설(Hypothesis)의 검증을 기반으로 연결하는 것을 의미한다. Rule 클래스에 맞는 객체를 만들어 Rule-chain에 활용한다. 각각 메소드로 구현이 되며, 각 체인의 이름을 메소드 명으로 봐도 무방하다. 체인은 'and' 혹은 'or' 조건으로 연결 될 수 있으며, 'and' 와 'or' 조건이 모두 사용될 수도 있다. 이러한 규칙의 연결을 통해 악성 URL 여부에 대한 정확도를 제시하고, 탐지의 정확성을 높이게 된다.

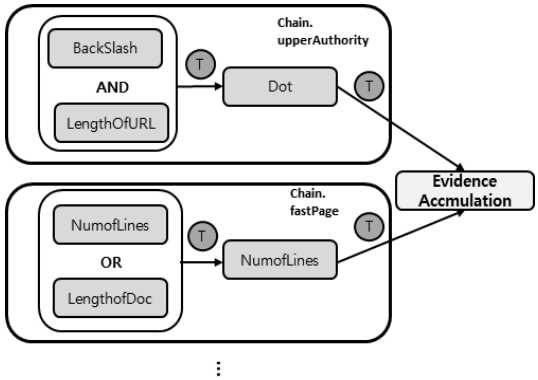

Fig. 4. Operation principle of Rule-chain

('and' condition)

Table 3.은 and 조건으로 연결된 Rule-chain의 명칭 및 설명을 제시한다. Chain1. upperAuthority는 URL 기반 Rule 클래스에 의해 생성된 backSlash, lengthOfURL, dot 객체를 통해 상위권한 탈취 가능성을 파악하기 위한 Rule-chain이다. '\'기호로 경로를 조작하고, '.'의 연속 사용으로 상위권한으로 이동할 수 있으며, 이와 같은 의심스러운 패턴을 긴 URL 주소를 통해 사용자가 알아차리지 못하게 한다는 것에 근거를 두고 생성하였다. 긴 URL 주소를 갖는다는 특성 자체가 악성 URL의 단독 기준으로 적용하기에는 약한 면이 있지만[9], backSlash, dot 클래스 객체와 연결되면, 악성 URL의 특징이 될 확률이 높아지게 된다. Chain2. SmishingUrl은 bitly 객체와 suspiciousWords 객체를 연결한 체인이다. 택배 스미싱 문자의 성행으로 인한 국내 택배사의 공식안

Table 3. Description of Chain 1.~5.

Rule-Chain Name	Description
upperAuthority	Malicious urls can capture upperauthority of system.
SmishingUrl	"Bitly" url form and suspicious words feature smishing url.
linkPossibility	Malicious url has uspicious link.
tagExist	Url has doubtful tags.
IPUrl	Malicious urls intend users to access specific IP address.

내문에서 착안한 것으로 택배사가 택배 관련 문자를 보낼 때 단축 URL을 사용하여 전송하는 일은 없다[12]. 그러므로 단축 URL 안의 HTML 내용에 '택배'라는 단어가 들어있음이 파악되면, 이는 택배 사이트를 가장한 피싱 사이트임이 명확해진다. Chain3. linkPossibility는 aTag 객체와 iframeTag 객체를 사용하였다. 이미지를 담는 〈iframe〉태그 안에 〈a〉태그의 링크가 걸려있다면, 보통 사이트 이미지를 캡처한 뒤, 사용자에게 정상적으로 동작하는 사이트처럼 보이게 하는 피싱 사이트일 확률이 높다[5]. Chain4. tagExist는 iframeTag, hiddenTag, unEqualHtmlTag 객체를 연결하여 의심스러운 태그들이 많이 포착된 사이트를 탐지하도록 설계하였다[5][6]. Chain 5. IPUrl은 dot 객체와 containDigits 객체를 연결하여 URL에 IP주소가 존재하는 경우를 탐지한다.[5]. Fig. 5.는 upperAuthority 체인을 구현한 코드이다.

```
public boolean upperAuthority(){
    if(backSlash.isHypothesis() && lengthOfURL.isHypothesis()){
        if(dot.isHypothesis()){
            return true;
        }
    }
    return false;
}
```

Fig. 5. Code of 'upperAuthority' Chain

('or' condition)

Table 4.는 or조건으로 연결된 Rule-chain의 명칭 및 설명을 제시한다. Chain 6. TLD는 ChinaTLD(.cn)와 RussiaTLD(.ru)를 or 조건으로 연결하여 접속 사이트의 국가를 기반으로 악성 URL인지를 판별할 수 있도록 하였다[10][11]. Chain 7. javaExe는 evalMethod, windowOpenMethod, charCodeAtMethod, escapeMethod, fromCharCodeMethod, parseIntMetod 객체를 or 조건으로 엮어 취약한 Java 함수의 존재 유무에 따라 취약한 사이트로 판별할 수 있게 하였다[5][6]. Chain 8. SpecialCharacter은 atMark, equalSign, hypen, semiColon 객체를 or 조건으로 엮어 URL 주소 자체에 특수문자가 있는 경우 의심스러운 사이트로 판별할 수 있게 하였다[5][6][7]. Fig. 6.은 TLD 체인을 구현한 코드이다.

Table 4. Description of Chain 6.~8.

Rule-Chain Name	Description
TLD	Malicious url from China or Russia.
javaExe	Malicious urls have vulnerable javascript functions.
SpecialCharacter	Malicious urls have special characters.

```
public boolean TLD() {
    if(tld.isCn_hypothesis()||
            tld.isRu_hypothesis()) {
        return true;
    }
    return false;

}
```

Fig. 6. Code of 'TLD' Chain

('and' & 'or' condition)

Table 5.는 and 및 or 조건으로 연결된 Rule-chain의 명칭 및 설명을 제시한다. Chain 9. fastPage는 Rule-set의 numberOfLines객체와 lengthOfDoc 객체를 or 조건으로 엮어 HTML 코드 내용의 길이를 체크하고, unequalHtmlTag 객체와 and 조건으로 엮어 최종적으로 빠르게 만들어진 피싱 페이지임을 확인할 수 있게 하였다[6].

Fig. 7.은 fastPage 체인을 구현한 코드이다.

Table 5. Description of Chain 9.

Rule-Chain Name	Description
fastPage	Malicious urls might be formed quickly.

```
public boolean fastPage(){
    if((numberOfLines.isHypothesis()||lengthOfDoc.isHypothesis())
            &&suspiciousWords.isHypothesis()){
        return true;
    }
    return false;
}
```

Fig. 7. Code of 'fastPage' Chain

V. HypoChain1.1.2 구성 요소 및 동작 원리

5장은 4장에서 살펴본 악성 URL 규칙 및 Rule-chian 을 바탕으로 HypoChain1.1.2 라이브러리의 전체 구성 및 동작 원리에 대해 제시한다. Fig. 8.은 HypoChain1.1.2 라이브러리의 전반적인 동작 원리를 나타내고 있다. 라이브러리 개발 환경은 Eclipse IDE를 활용했으며, 이는 크게 4개의 클래스 및 외부 라이브러리로 나눌 수 있다.

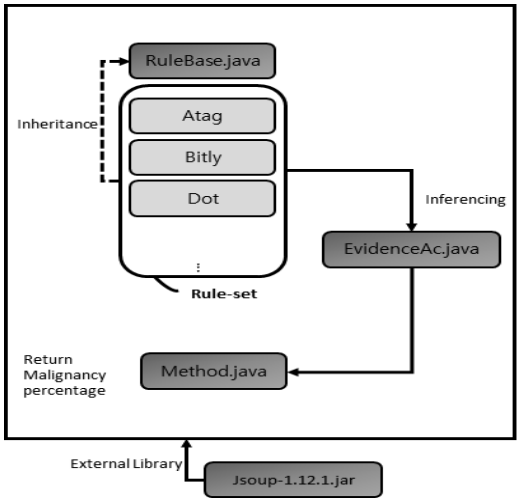

Fig. 8. Operation principle of HypoChain

5.1 RuleBase 클래스

RuleBase 클래스는 악성 URL의 판단 기준이 되는 Rule들의 기본 토대를 제공한다. Rule-set에 포함되어있는 각각의 기준 클래스들은 RuleBase 클래스를 상속받아 rule()메소드를 오버라이딩 한다.

5.2 Rule-set 및 Rule 클래스

현재 HypoChain1.1.2의 악성 URL 탐지에 사용되는 Rule-set에는 총 23개의 Rule 클래스들이 구성되어 있다. rule()메소드의 조건을 만족하게 되면, setHypothesis()라는 함수를 통해 hypothesis변숫값을 'true'로 설정한다. isHypothesis()함수는 최종적으로 설정된 hypothesis 결괏값을 반환해주는 역할을 한다.

5.3 EvidenceAc 클래스

EvidenceAc 클래스는 HypoChain의 검증 엔진 역할을 한다. 객체화한 Rule 클래스의 rule() 메소드를 호출하여 입력받은 URL이 조건에 만족하는지 확인한다. 또한 Rule들을 'and' 혹은 'or' 조건으로 엮어서 여러 개의 Rule-chain을 형성하였다. Rule-chain 메소드는 결괏값을 'true' 혹은 'false'로 반환한다.

5.4 Method 클래스

라이브러리가 적용된 환경에서 Method 클래스의 checkURL() 함수를 호출하면 최종적인 확률값이 반환된다.

5.5 외부 라이브러리 (Jsoup-1.12.1)

HypoChain1.1.2의 외부 라이브러리로는 Jsoup-1.12.1.jar 파일을 포함한다. Jsoup 라이브러리는 URL의 HTML 구문을 분석하고, 데이터를 추출하는 기능을 제공하고 있다[4].

라이브러리를 이용하려는 개발자는 기존에 구현되어 있는 Rule 클래스들의 코드를 재사용할 수 있다는 장점이 있다. 또 Rule을 추가하고 싶은 경우엔 RuleBase 클래스를 상속하여 체계에 맞게 구현할 수 있다.

VI. 탐지 규칙 추가/삭제의 용이성 및 동적인 Rule-chain 구성

본 연구는 기존의 악성 URL의 특징에 대한 신뢰성 있는 이론 연구를 바탕으로 한다는 가정하에 기술을 개발하였다. 개발 결과를 통해 '탐지규칙 추가의 용이성' 및 'Rule-chain 생성 통한 효과성'에 대해 제시하고자 한다.

6.1 탐지 규칙 추가의 유연성

Rule-set에 규칙들을 클래스화 하여 생성함으로써, 새로운 탐지규칙이 발견됐을 시, 이를 반영하기가 간단해진다. 만약, 최신 악성 사이트의 새로운 URL-based 특징이 발견되었다고 가정한다면, 규칙 추가는 다음과 같이 이루어질 수 있다.

Rule-set에 NewRule.java라는 자바 클래스를 추가로 생성하여, RuleBase 클래스를 상속받는다. Fig. 9.와 같이 일종의 템플릿처럼 변수 및 가설설정 함수를 구성하고, rule() 메소드 부분에만 해당 특징을 코드로 구현해주면 새로운 규칙이 생성된다.

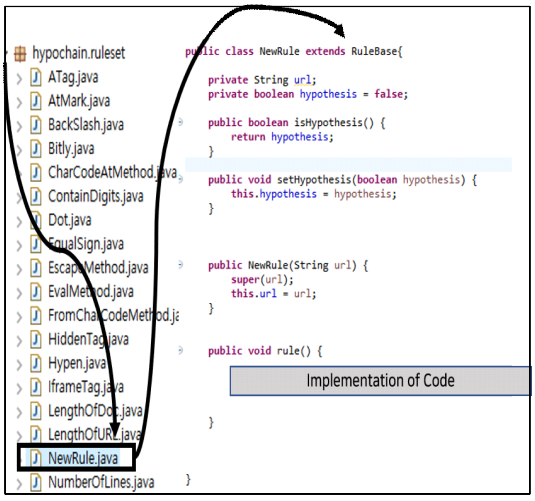

Fig. 9. Method of adding new Rule

6.2 새로운 규칙에 대한 Rule-chain 구성

새로운 규칙이 추가되면 해당 규칙이 검증해주는 가설이 생성된 것이고 만일 이 가설이 '참'이라면 악성 URL을 탐지하는데 활용되는 새로운 증거가 된다. 이러한 구성은 악성 URL의 탐지를 검증된 가설을 기반으로 Rule의 Chain을 구성할 수 있게 하여 규칙의 추가가 전체 악성 URL 탐지에 효과적으로 반영되도록 한다.

VII. Android 환경적용 결과

7.1 라이브러리 구현

Rule-set 체계를 통해 구축된 HypoChain 프로젝트를 java 라이브러리화하여 '.jar' 형식으로 포맷팅한다. 이렇게 하여 생성된 HypoChain1.1.2 라이브러리 파일을 Android 환경에 적용하여 테스트 프로젝트를 생성하였다.

HypoChain1.1.2 라이브러리에 포함되어 있는 Jsoup라이브러리를 이용하여 입력된 URL의 HTML 데이터를 가져오는 작업을 진행한다. Jsoup 라이브러리의 JsoupAnsytTask 클래스 객체를 생성하고 execute()메소드를 실행한다. JsoupAsyncTask 클래스의 doInBackground()메소드에서 HypoChain1.1.2 라이브러리를 사용하기 위해 method 클래스의 정적(static) 함수 checkURL 메소드를 호출한다. checkURL 메소드는 url의 주소 정보와 HTML의 악성 가능성을 비율 값으로 리턴한다.

```
private class JsoupAsyncTask extends AsyncTask<Void, Void, Void> {

    private int percent = 0;

    @Override
    protected void onPreExecute() { super.onPreExecute(); }

    @Override
    protected Void doInBackground(Void... params) {
        percent = method.checkURL(url);
        return null;
    }

    @Override
    protected void onPostExecute(Void result) {

        test.setText(percent + "%");


    }

}
```

Fig. 10. Code of Calling HypoChain checkURL() method

7.2 테스트 결과 화면

Fig. 11.은 악성 사이트 탐지율 출력 화면의 예시를 보여준다. Fig. 10.와 같은 결과를 나타내는 onPostExecute()메소드는 최종 URL의 악성 확률값을 TextView로 표출한다. 첫 번째 테스트 결과, "https://comic.naver.com/index.nhn" URL의 악성 탐지 정도는 0%로 출력이 되었다.

"http://img5-upload.xiaoyouxi.cn/mm/101/qq.html"의 중국 URL을 두 번째 테스트에 적용해본 결과 URL의 악성 탐지 정도는 33%로 출력이 되었다.

출력된 확률값은 다음 수식(1)과 같은 계산을 통해 검증된 결과이다. 수식에서 전체 Rule-chain의 개수를 변수 T, true 값을 반환하는 Rule-chain의 개수를 변수 t라 한다.

$$percentage(\%) = \frac{t}{T} * 100 \qquad (1)$$

따라서, 0%의 결과는 해당 URL이 총 9개의 Rule-chain 중 0개의 Rule-chain을 만족했다는 의미이고, 33%의 결과는 총 3개의 Rule-chain에 대해 충족되었다고 볼 수 있다.

추후 Rule 추가에 따른 Rule-chain의 조합이 다양해질수록, 반환된 확률값은 더 정확해질 수 있다.

Fig. 11. Outputs of Malicious Probability

VIII. 결 론

본 논문에서의 HypoChain 라이브러리 개발 및 Android 환경에서의 적용 결과 다음과 같이 결론을 내릴 수 있다.

HypoChain 라이브러리는 악성 사이트 판별을 위한 Rule들을 클래스 별로 독립시켰다. 기준을 추가하고 싶을 때 클래스를 생성하여 구현할 수 있고, 따라서 최신 악성 사이트의 특징들을 바로 하나의 클래스로 구현할 수 있다는 장점이 있다. 3장의 통계 결과를 통해, 사용자의 행위적 특성을 고려하여 메신저 어플 등에 직접 탐지 기능을 탑재하는 것이 어느 정도 효과가 있는지 가늠하였다. 개발자는 어플 설계 시, 보안 기능 구현을 위해 타 회사의 보안 솔루션에 의지할 필요 없이 HypoChiain 라이브러리를 이용해 본인이 직접 기준을 구현하고 어플에 적용할 수 있다.

HypoChain은 현재 데모 버전으로 1.1.2버전 개발단계에 있지만, 추후 탐지율을 높이기 위해 최신 악성 사이트의 동향을 파악하여 Rule을 추가할 계획이다. 버전을 업그레이드하여 탐지율을 높인 다음 Github과 같은 코드 공유 사이트에 라이브러리 파일을 올려 무료로 배포할 계획이다.

References

〔1〕 Dongwon Seo, Arindam Khan, and Heejo Lee, "A study on detecting malcodes distribution sites," Korean Information Processing Society, 15(2), pp. 1-4, Nov. 2008

〔2〕 Jiho Cho, "A study on smishing detection technique using taintdroid," Convergence Security Journal, 15(1), pp. 3-9, Feb. 2015

〔3〕 Ho-Kyoung Shin, Ji-Myoung Shin, and Ho Lee, "A study on the factors influencing information sharing in the social network services," Journal of Information Management, 42(1), pp. 137-156, Jan, 2011

〔4〕 Java Jsoup Library, "Jsoup Library", https://www.jsoup.org/, Mar. 7. 2020

〔5〕 Core, "Phishing websites features", https://core.ac.uk/download/pdf/30732240.pdf, Mar, 2020

〔6〕 Wan Nurulsafawati Wan Manan, Abdul Ghani Ali Ahmed, and Mohd Nizam Mohmad Kahar, Intelligent Computing& Optimization, pp. 210-218, Springer International Publishing, Jan, 201

9
〔7〕 Ram B.Basnet, Andrew H. Sung, and Quingzhong Liu, “Learning to detect phishing urls,” International Journal of Research in Engineering and Technology, vol. 3, no. 6, pp. 11-24, June, 2014
〔8〕 Yong-bum Cha, “Proactive response against smishing using shorten url,” Journal of Digital Forensics, 9(1), pp. 19-31, June, 2015
〔9〕 Ji-sun Park, “Based on url pattern analysis preventive measures against harmful sites,” Master’s Thesis, Konkuk University, Nov. 2018
〔10〕 KISA, Malware Hidden Site Detection Trend Report〔The first half of 2019〕, Jul, 2019
〔11〕 KISA, Malware Hidden Site Detection Trend Report〔The second half of 2019〕, Jan, 2020
〔12〕 CJLOGITICS, “CJ Parcel Service”, https://www.cjlogistics.com/ko/newsroom/notice/NR_00000010, Mar. 7, 2020
〔13〕 Jeong-Wu Ha, Hyu-Kang Kim, Jong-in Lim, “Research on malicious code hidden website detection method through whitelist-based malicious code behavior analysis,” Journal of The Korea Institute of Information Security and Cryptology(JKIISC), 21(4), pp. 61-75, Aug. 2011
〔14〕 Jae-Kyung Park, “A study of realtime malware url detection & prevention in mobile environment,” Journal of The Korea Society of Computer and Information, 20(6), pp. 37-42, June. 2015
〔15〕 Anhlab, "V3 mobile security," https://www.ahnlab.com/kr/site/product/productView.do?prodSeq=19, Mar 25, 2020
〔16〕 Pill App, “EST Security”,https://www.estsecurity.com/, Mar. 26, 2020

〈저자소개〉

김 보 민 (Bo-Min Kim) 학생회원
2017년 3월~현재 : 서울여자대학교 정보보호학과 학사과정
2019년 3월~현재 : 서울여자대학교 정보보호학과 학부연구생
〈관심분야〉 정보보호, 모바일 보안, 클라우드 보안

한 예 원 (Ye-Won Han) 학생회원
2017년 3월~현재:서울여자대학교 정보보호학과 학사과정
〈관심분야〉 정보보호, 암호학, 네트워크 보안, 시스템보안

김 가 영 (Ga-young Kim) 학생회원
2017년 3월~현재:서울여자대학교 정보보호학과 학사과정
〈관심분야〉 정보보호, 암호학, 모바일보안

김 예 분 (Ye-Bun Kim) 학생회원
2017년 3월~현재:서울여자대학교 정보보호학과 학사과정
〈관심분야〉 정보보호, 인공지능, 암호학

김 형 종 (Hyung-Jong Kim) 종신회원
1996년: 성균관대학교 정보공학과 공학사
1998년: 성균관대학교 정보공학과 공학석사
2001년: 성균관대학교 전기전자 및 컴퓨터공학과 공학박사
2001~2007년: 한국정보보호진흥원 수석연구원
2004~2006년: 미국 Carnegie Mellon University, CyLab 국제공동연구원
2013~2014년: 미국 Carnegie Mellon University, ECE, Visiting Professor
2007~현재: 서울여자대학교 정보보호학과 정교수
〈관심분야〉 안드로이드 환경의 IoT서비스 개인정보보호, 블록체인 서비스 성능평가, 클라우드 서비스 보안 모델