DID인증기술_CapStoneDesign

Apache Access.log와 Yara Library를 활용한 웹쉘 탐지

• 지도 교수 : 신 승 수

• 학　　과 : 정보보호학과

• 팀　　명 : 술 래

목 차

그림 목차

표 목차

Ⅰ. 서론

1. 연구 배경

웹의 대중화 시대로 들어선 21세기 시점에서 웹 보안에 대한 관심은 필연적으로 고조되고 있다. 또한 보안 위협 중 웹 기반 해킹 공격이 절반 이상의 높은 비중을 차지하며, 그 중 웹쉘을 통한 웹 서버 공격이 주를 이루고 있다.

웹 취약점을 이용하고 다양한 탐지 우회기법이 적용된 웹쉘은 일반적인 서버관리자가 해킹여부를 확인하기 힘들고, 피해를 인지하더라도 탐지가 어려워 대응의 한계가 명확하다[1].

따라서 오픈 소스 기반의 웹쉘 파일들에 대한 분석과 그에 따른 적절한 탐지방안의 적용이 필요하다.

2. 연구 목적

방화벽, IPS, IDS 등 다양한 네트워크 보안장비를 갖추고도 웹 서버 안에 미리 숨겨져 있던 웹쉘로 인해 보안 사고가 발생하는 경우도 있다.

웹쉘을 탐지하는 솔루션으로는 방화벽이나 백신을 사용하기 보다 웹쉘 전용 보안 솔루션을 사용하는 것이 효과적이다. 방화벽은 한 번 허용된 IP, PORT로 부터 오는 공격을 방어하지 못하고, IDS/IPS는 네트워크 계층에서 동작하여 어플케이션 계층에서 작동하는 웹쉘을 탐지하기는 어렵다[2].

이러한 문제를 해결하기위해 본 논문은 Apache Access.log와 Yara Library를 활용한 웹쉘 탐지 프로세스를 제안한다.

Ⅱ. 관련 연구

1. 웹쉘

웹쉘은 해커가 악의적인 목적으로 웹 서버에서 임의의 명령(파일 탐색, 시스템 쉘 명령)을 실행할 수 있도록 제작한 스크립트 파일을 의미한다. 다양한 방법으로 만들 수 있지만, Server Side Script(JSP, PHP, ASP…)로 만드는 방법을 자주 사용한다.

해당 스크립트는 웹 서버의 취약점을 통해 웹 서버에 업로드되며 해커가 시스템에 접근하여 시스템 제어 및 서버에 악성코드를 설치해 사용자의 PC를 공격하거나 연결된 데이터베이스의 정보를 유출하는 등 큰 피해를 줄 수 있다[3].

가. 웹쉘 탐지 우회기법

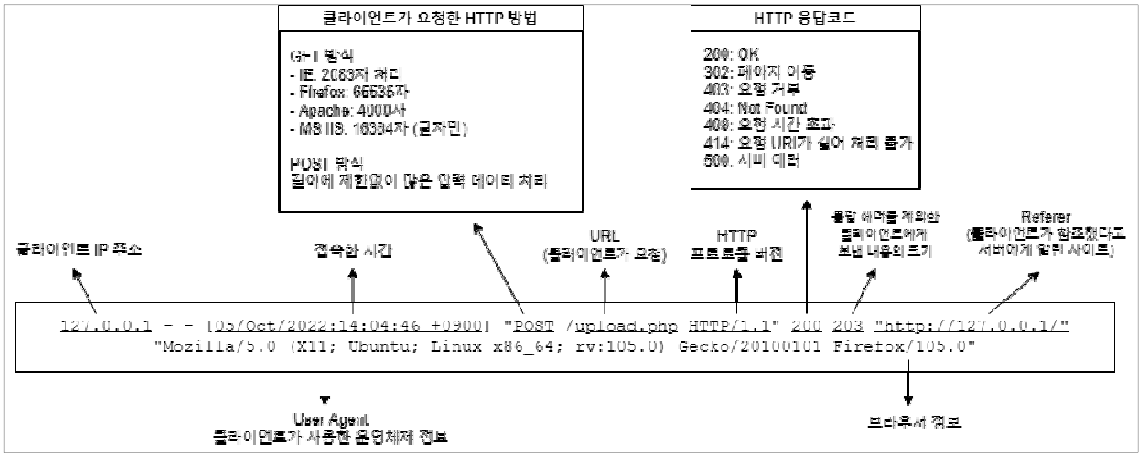
웹쉘은 한줄 웹쉘, 인코딩, 문자열 분리, 파일 생성, HTTP 서비스와 같은 기법을 사용하여 웹쉘 탐지 솔루션의 탐지를 우회한다. 또한 일반적인 서버 관리자들은 웹쉘을 통한 해킹의 피해여부 조차 파악하기 힘든 것이 현실이다.

나. 기존 탐지 방식 및 한계점

현재 상용화 되어있는 웹쉘 탐지 솔루션들 중 대부분은 정규표현식에 기반한 패턴 탐지 기반의 솔루션이 대부분이지만 시그니처 기반의 솔루션(Firewll,IPS,WAF)으로는 신종 웹쉘, 보안장비 우회기법, 등 신규 웹쉘에 대한 방어가 현실적으로 어렵다. 시그니처 기반의 진단 방식은 샘플을 수집하고 진단하는 방식으로, 기존에 발견된 웹쉘에 대한 탐지는 즉각적으로 가능하나 신/변종 웹쉘에 대한 탐지는 불가능하기 때문이다[2].

2. Apache Access.log

Access.log는 서버로부터 HTTP 프로토콜을 통하여 접속자에 관한 전송된 정보들을 보여주는 역할을 하고 서버에 전송 및 저장되는 정보 및 저장되는 형식은 [그림 2-1] 과 같다. 해당 파일에 수집되는 로그는 HTTP 통신을 사용하며 HTTP header와 HTTP body로 구성되어 있다. 그 중 HTTP 헤더의 요청 헤더에는 요청한 URL과 메소드 (GET, POST), 요청 생성에 사용된 브라우저 및 기타 정보 등이 포함되지만 개발 환경에 따라 POST 방식의 로그 수집이 진행되지 않는 경우가 발생한다.



[그림 2-1]. Apache Access.log 구성도

먼저 GET 방식은 데이터를 요청하기 위한 자원을 전송할 때 URL 뒤에 “?” 마크를 통해 URL과 데이터 표현을 구분하며 데이터를 전송하는 명령을 파라미터에 표시하게 된다. 반면에 POST 방식은 전송할 데이터를 HTTP Body에 담아서 전송하기 때문에 파라미터에 표시되지 고, 로그 수집도 진행되지 않는다[4].

공격 로그를 파싱하는 Access.log 파일은 GET 방식의 로그만 수집하지만 최근 공격에 사용된 대부분의 웹쉘은 POST 방식을 채택하고 있다. 본 논문에서는 이를 해결하기 위해 Apache 환경에서 사용하는 dumpio 메소드을 사용한다[5].

3. Yara Library

Yara Library는 서버 및 웹에 침투한 악성 파일의 행위 특징을 정의하여 악성 코드의 종류들을 식별하고 분류하는 목적으로 사용하는 도구이다.

악성 코드를 탐지하기 위한 방법으로 텍스트 문자열 및 바이너리 패턴을 사용한다. 이러한 이유는 악성코드를 이용한 행위의 특징은 파일, 프로세스에 포함된 텍스트 문자열 또는 바이너리 패턴에서 나타나기 때문이다[6].

본 연구에서는 Apache Access.log 중 URL 데이터와 Yara Libray의 텍스트 문자열 패턴 매칭을 이용하여 Rule을 정립하고 이를 바탕으로 서버 내 동작중인 웹쉘의 패턴을 탐지하여 파일을 식별하고 제거 또는 분류한다.

Ⅲ. 웹쉘 탐지 프로그램

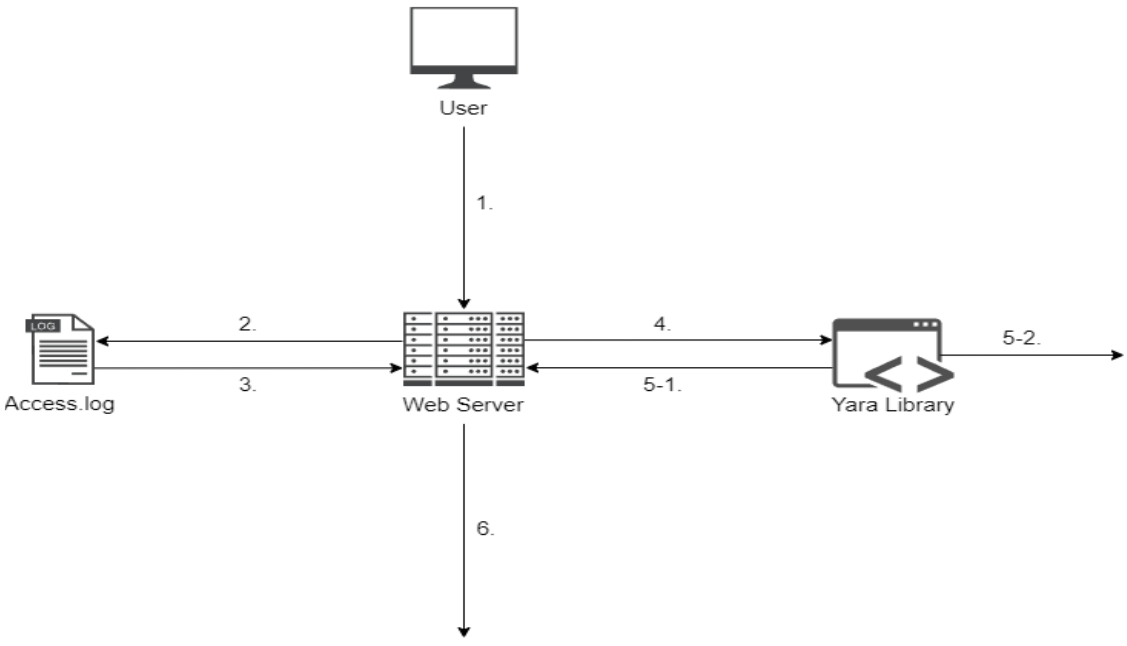
본 절에서는 프로그램의 구성과 프로그램에서 수행하는 기능들의 진행과정, 설계 방법에 대해 기술한다.

1. 탐지 프로그램 설계

가. 프로그램 구성

제안하는 프로그램은 Apache Access log와 Yara Library, GUI로 구성된다. Apache Access.log는 서버에 접근하는 사용자들의 기록을 수집하며, 클라이언트 주소, HTTP, URL 등으로 구성된다.

본 연구에서는 log에 저장되는 URL 데이터의 ‘실행 파일의 절대 경로’, ‘실행 파일의 파일명’ 그리고 ‘사용자의 입력값’를 이용하여 파싱 및 저장한다. Yara Library는 파일의 시그니처를 탐색하여 악성코드의 종류를 식별하고 텍스트 문자열을 탐지하기 위한 Yara Rule을 정립하는 역할을 한다. 또한, 프로그램은 사용자의 가시성을 위해 GUI로 구성된다. 프로그램의 전체 구성도는 [그림 3-1]과 같다.



[그림 3-1]. 웹쉘 탐지 프로그램 구성

나. 프로그램 설계

본 논문에서 제안된 웹쉘 탐지 프로그램의 주요 기능 설계는 GUI, Apache Access log와 Yara Library로 구성된다. GUI는 사용자들이 직관적으로 조작 방법을 이해하고 프로그램을 사용할 수 있도록 탐지, 파일 제거, 종료 인터페이스를 설계한다. Apache Access log는 공격자가 공격 구문을 입력하여 웹쉘의 실행 유무를 확인하기 위해 사용하는 것으로 정규표현식을 통해 패턴을 찾아내야 하므로 Python을 통해 파싱하는 프로그램을 설계한다. 도출된 결과는 텍스트 파일로 저장한다.

Yara Library는 공격자가 사용하는 시스템 명령어를 특정하기 위해서 Yara Library를 활용하여 Yara Rule을 정립한 후 저장된 텍스트 파일에 적용하도록 설계한다. 이때 도출된 결과를 바탕으로 웹쉘 미탐지 파일은 정상 파일로 판단하고 유지, 웹쉘 탐지 파일은 격리한 후 삭제할 수 있도록 설계한다.

2. 탐지 프로그램 구현

가. 프로그램 구현 환경

구현 환경은 우분투로 구현하며 Apache Server와 Tomcat을 사용한다.

<표 3-1>. 프로그램 구현 환경

항목	버전
Ubuntu	Ubuntu-18.04
Apache Server	Apache/2.2.22
Tomcat	Tomcat/9.0.68

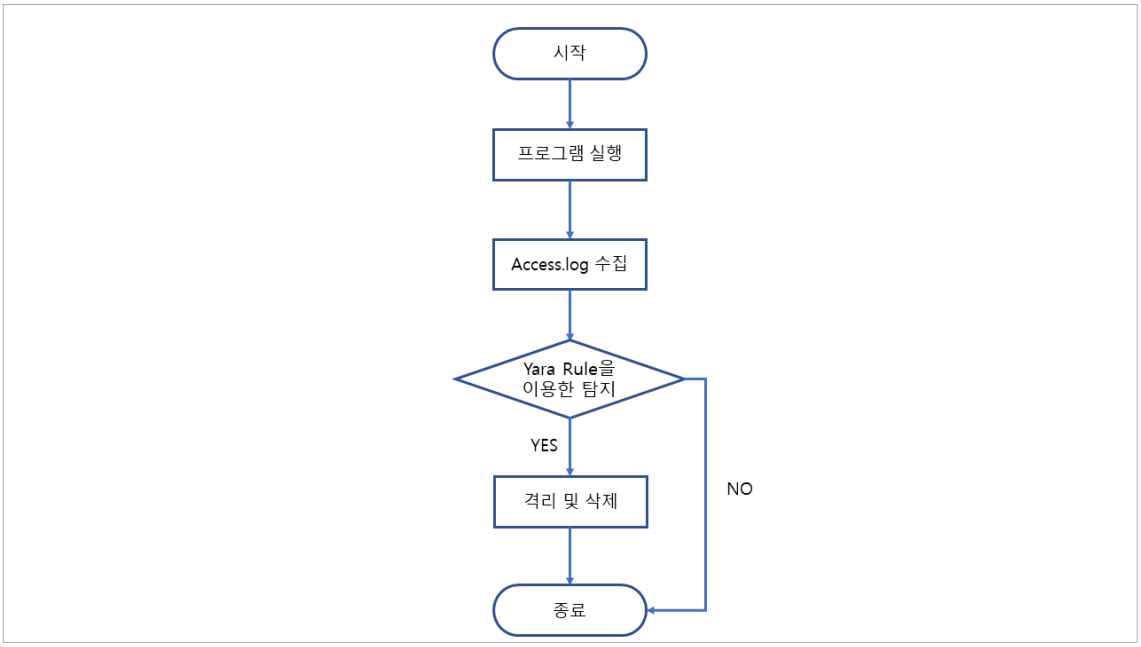
나. 프로그램 동작 과정

사용자가 프로그램을 실행한 후 탐지된 결과를 확인하거나 웹쉘이 삭제 되는 과정을 사용자 관점과 프로그램 구성요소별 관점으로 나누어 각각 프로그램 순서도와 프로그램 흐름도를 사용하여 기술한다.

(1) 프로그램 순서도

사용자가 프로그램을 실행하면 웹 서버에 업로드되어 있는 웹쉘 코드를 탐지하고 탐지된 웹쉘 코드를 삭제 및 격리하기까지의 동작 과정을 사용자의 관점에서 시스템 순서도를 사용하여 기술한다.

사용자는 프로그램 실행하여 웹쉘을 유무를 확인하고, 탐지된 웹쉘이 있다면 삭제 및 격리 조치를 한다.

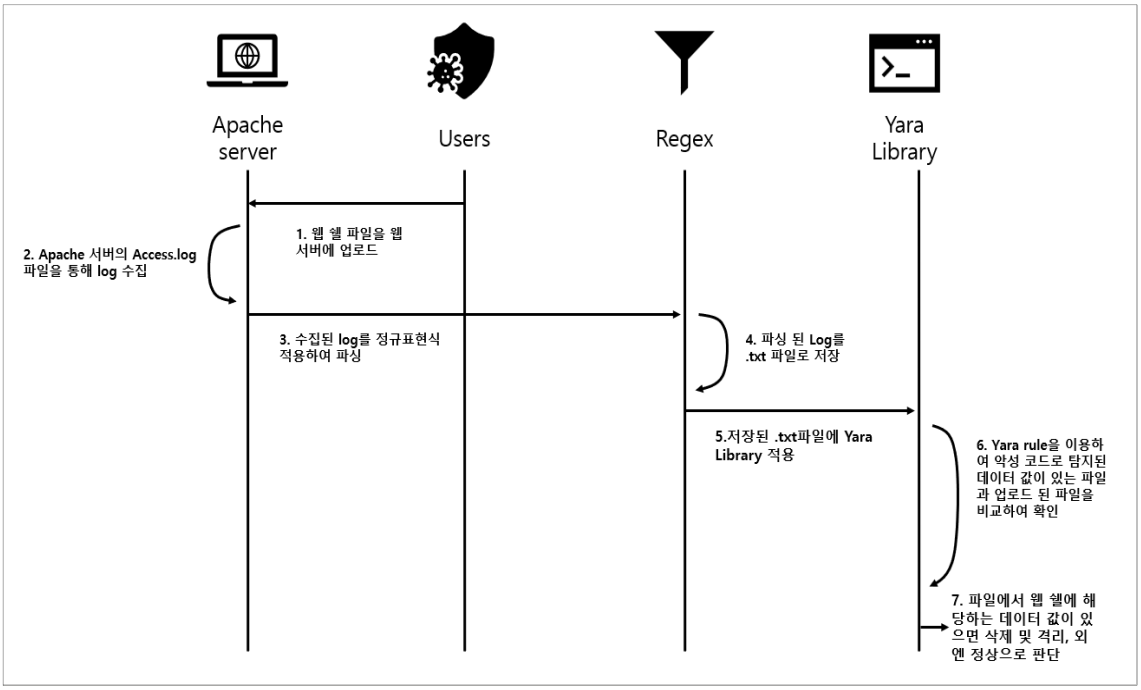


[그림 3-2]. 웹쉘 탐지 프로그램 순서도

(2) 시스템 흐름도

사용자가 프로그램을 실행한 후 웹쉘 탐지 프로그램을 탐지하여 삭제하는 동작 과정을 시스템의 구성요소별 관점에서 시스템 흐름도를 사용하여 기술한다.

사용자가 웹쉘 탐지 프로그램을 실행하면 Apache 웹 서버에 있는 Access log 파일에 생성된 로그 중 URL 데이터값만 정규표현식을 이용하여 파싱한다. 파싱된 데이터는 텍스트 파일로 저장하여 Yara Rule에 적용한다. 이후 도출된 결과를 바탕으로 정상적으로 탐지가 되었다면 파싱된 데이터 중 파일의 경로 및 파일명을 이용하여 탐지된 파일을 웹쉘로 분류시켜 파일을 삭제 및 격리하고 해당 과정에서 탐지가 되지 않는다면 정상 파일로 판단하고 유지한다. 자세한 순서도는 [그림 3-3]과 같다.

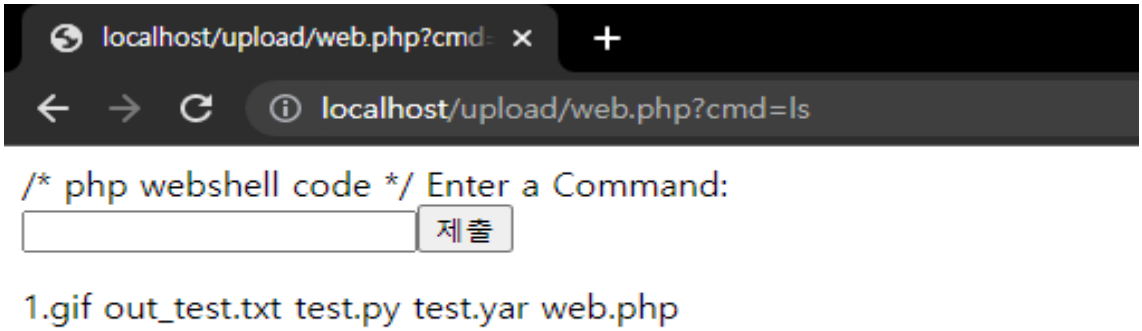


[그림 3-3]. 웹쉘 탐지 프로그램 흐름도

다. 프로그램 구현

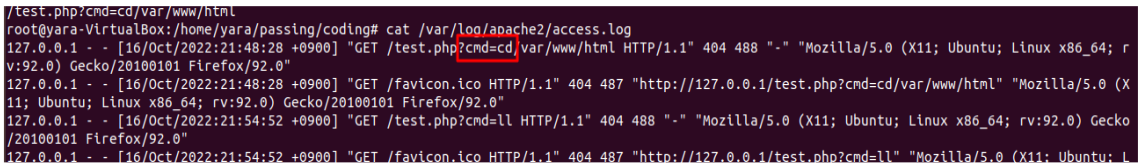
(1) 웹쉘 공격 예시 및 Apache Access.log

[그림 3-4]은 공격자가 웹 서버상에 업로드 된 웹쉘 파일을 통해 “cmd=ls” 입력 값을 통해 ls라는 명령어를 실행한 후 어떠한 파일들이 디렉터리 내에 포함되어 있는지 알아내는 과정이다.



[그림 3-4]. 웹쉘 공격 예시

해당 웹쉘 파일을 활용하여 공격한 후 PHP 기반의 Apache 웹 서버 Access.log에 해당 공격 행위 유무를 확인한다.



[그림 3-5]. Apache Access.log 파일에 남아있는 공격 로그

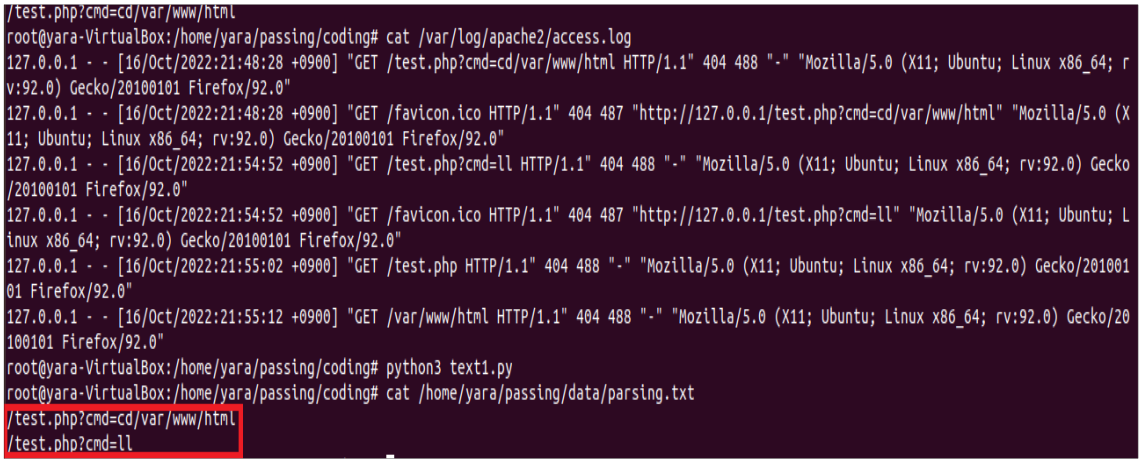
(2) 정규표현식을 이용하여 Access.log 파싱

우선 Access log에 포함되어 있는 수많은 로그 중 웹페이지가 업로드 되어있는 경로와 URL에 입력한 '공격자의 행위'에 대한 내용을 Python을 이용한 정규표현식으로 파싱한다. 파싱을 위해 사용된 정규표현식은 [그림 3-6]과 같다.

```python
import sys
import re

with open('/var/log/apache2/access.log') as fin:  # accesslog 파일 열고 읽기
  accessLogs = fin.readlines()

fout = open("/home/yara/parsing/data/parsing.txt", "w")  # 파싱한 로그 저장할 txt 파일 생성

for line in accessLogs:
  splitLine = line.split()  # accesslog 라인별로 ' '기준으로 구분
  userUrl = splitLine[6]  # 6번째가 url이므로 userUrl 변수에 저장
  reUrl = r'\/.+\?.+'  # url 형식 [ /절대경로/파일명?명령어 ]

  if re.match(reUrl, userUrl) != None:  # 정규표현식에 매치해보고 None값이 아니라면 txt 파일에 쓰기
    fout.write("{}\n".format(userUrl))
  else:  # None 값이라면 pass
    pass

fin.close()
```

[그림 3-6]. Apache Access.log 파싱 코드

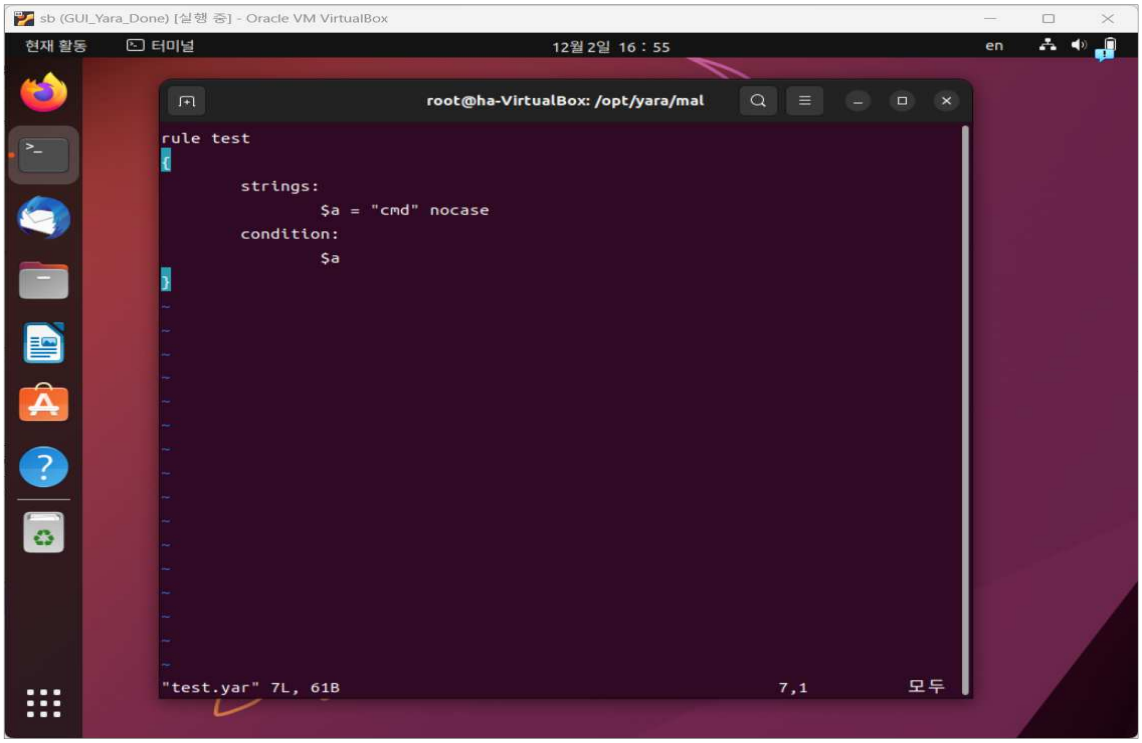
해당 정규표현식 내용이 포함되어 있는 Python 파일을 실행시키면 다음 [그림 3-7]과 같이 공격자가 사용한 웹쉘 파일명과 파일의 경로값을 얻어낼 수 있다.



[그림 3-7]. Apache Access.log 파싱 결과

(3) 웹쉘 탐지를 위한 Yara Rule 선언

정규표현식을 통해 파싱한 로그에서 웹쉘 공격을 탐지하기 위해 Yara Library를 활용하여 [그림 3-8]과 같이 Yara Rule을 정립하였다.



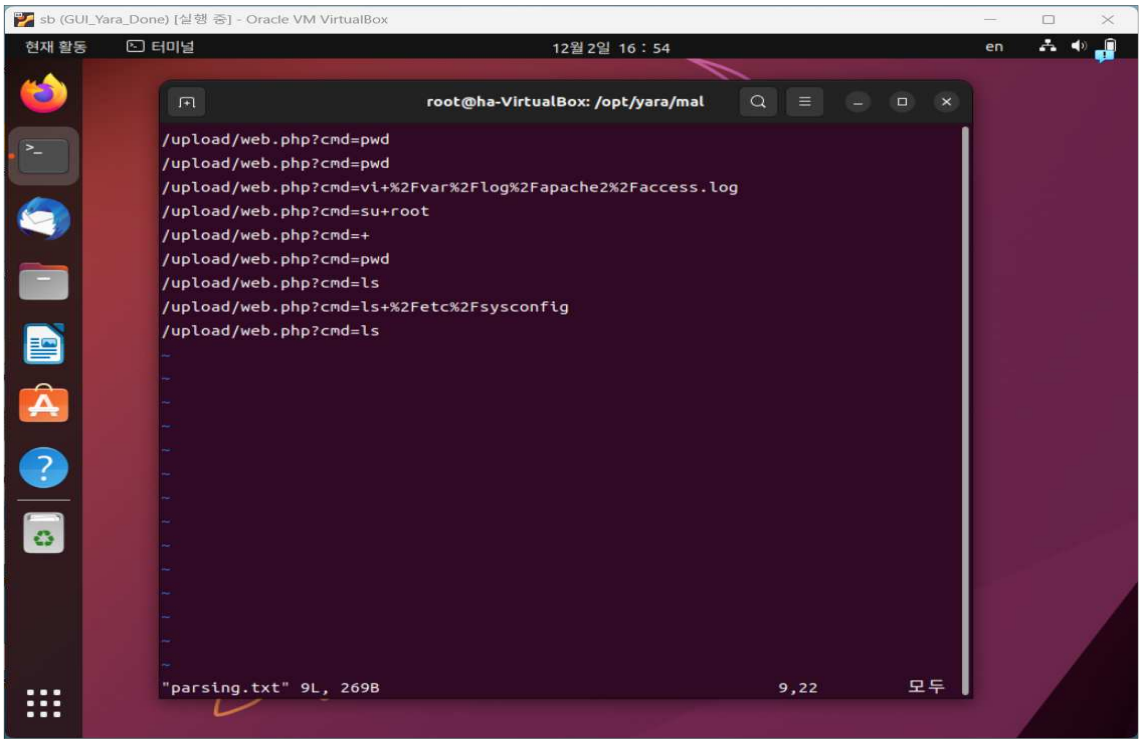
[그림 3-8]. Yara Rule 구성 코드

Rule은 Strings(문자열) 탐지 방식을 통해 로그 내에 “cmd” 라는 문자열이 포함되어 있는지 확인한다. 해당 Yara Rule을 통해 파일을 검사하여 웹쉘인지 아닌지에

대해 판단하는 방식이다.

(4) Yara Rule을 이용하여 웹쉘 탐지

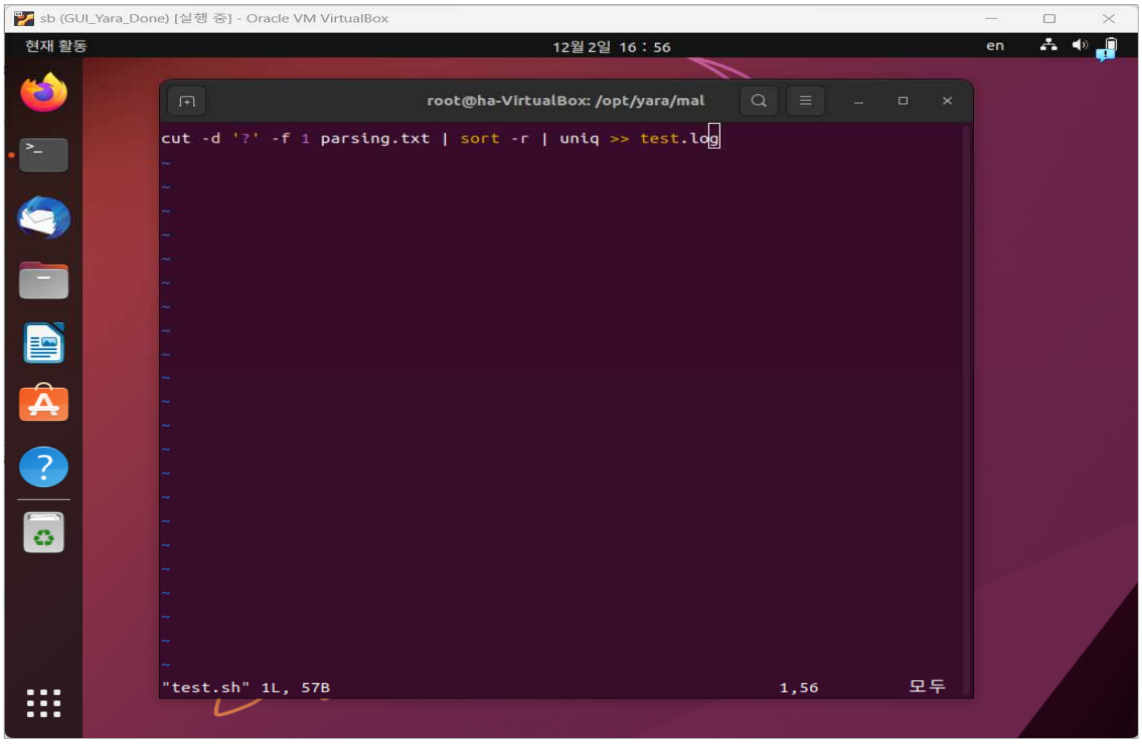
Yara Rule을 적용한 결괏값을 보면 “cmd” 문자열이 사용된 로그가 탐지되었음을 확인할 수 있다. 탐지 결과는 [그림 3-9]과 같다.



[그림 3-9]. 로그 내의 악성 행위를 탐지한 Yara Rule

(5) 탐지된 웹쉘 파일에 대한 삭제 또는 격리

탐지된 웹쉘을 삭제 또는 격리 조치를 하기 위하여 공격자가 공격에 사용한 웹쉘 파일의 절대경로가 필요하다. 절대경로 또한 정규표현식을 통해 웹쉘의 절대경로를 파싱하여 얻어낸다. 절대경로를 획득하는 정규표현식은 [그림 3-10]과 같다.

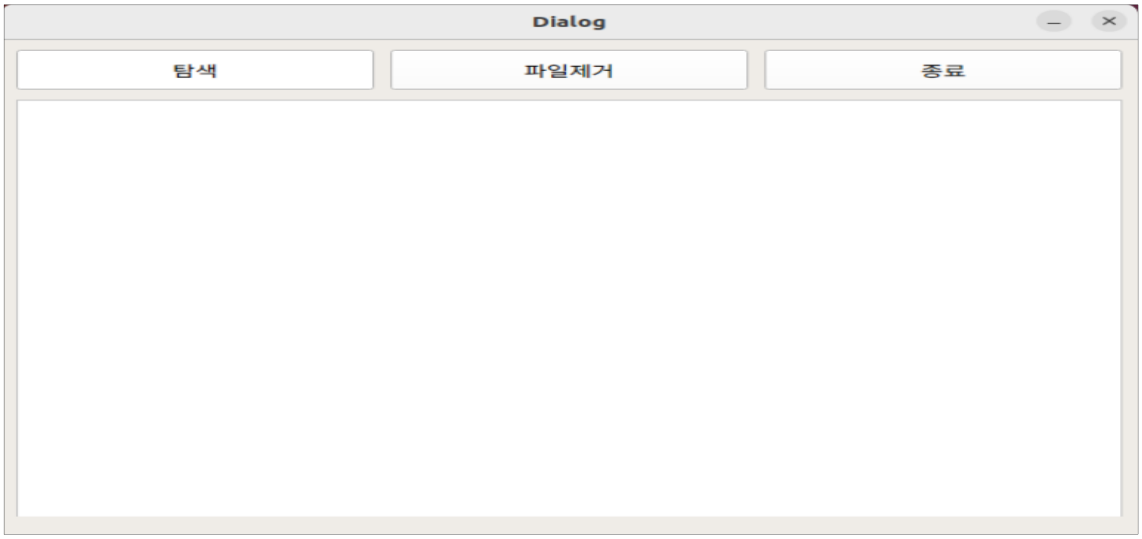


[그림 3-10]. 탐지된 웹쉘의 절대경로 획득

획득한 파일의 절대경로는 Python 코드를 이용하여 삭제 또는 격리를 진행한다.

(6) GUI 구현

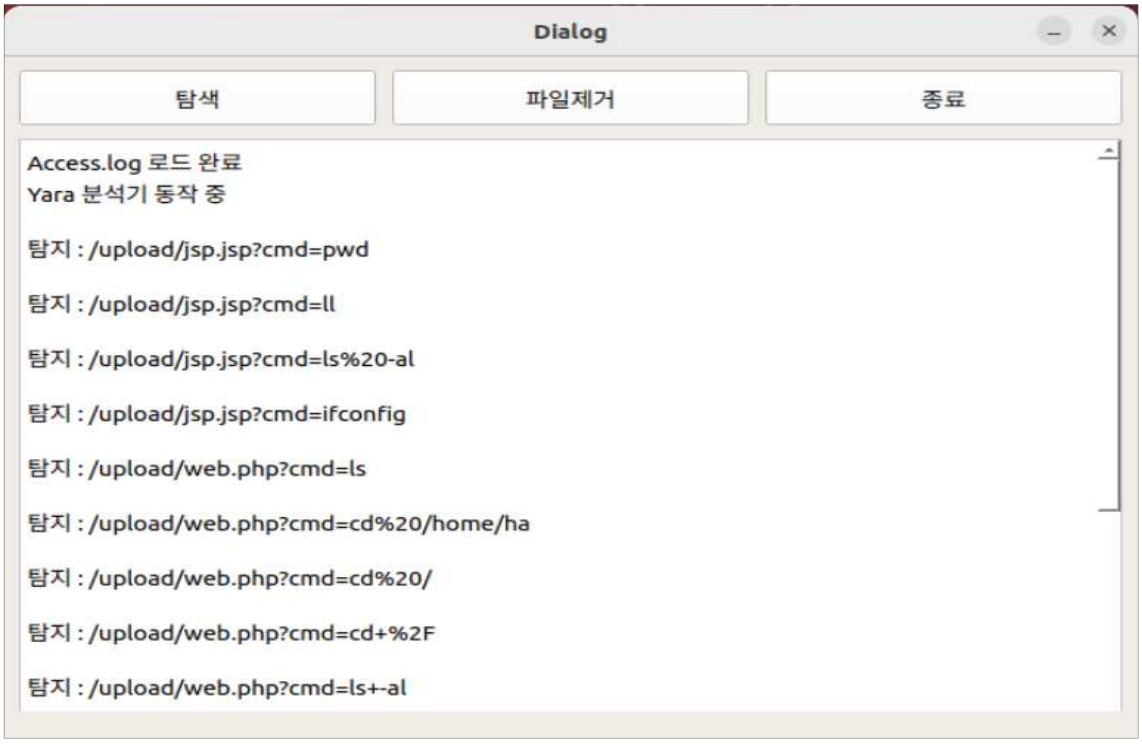
개발된 프로그램의 GUI 실행화면은 다음 [그림 3-11]과 같다.



[그림 3-11]. GUI 실행화면

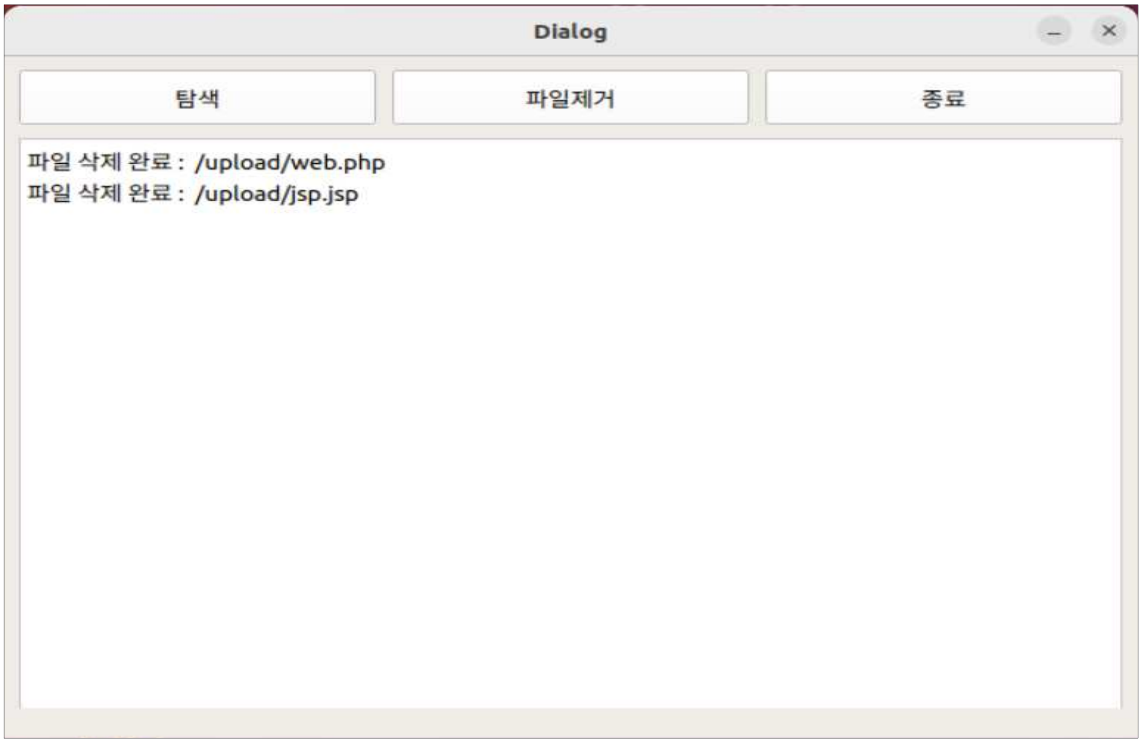
프로그램을 실행시키게 될 경우 총 3개의 버튼으로 구성된 초기 화면을 볼 수 있다. 각각 탐색(웹쉘 탐지), 파일제거(탐지된 웹쉘 파일 제거), 종료(프로그램 종료)의 기능으로 구성되어 있다.

탐색 버튼을 통한 이벤트를 실행시킨 결과는 [그림 3-12]과 같다.



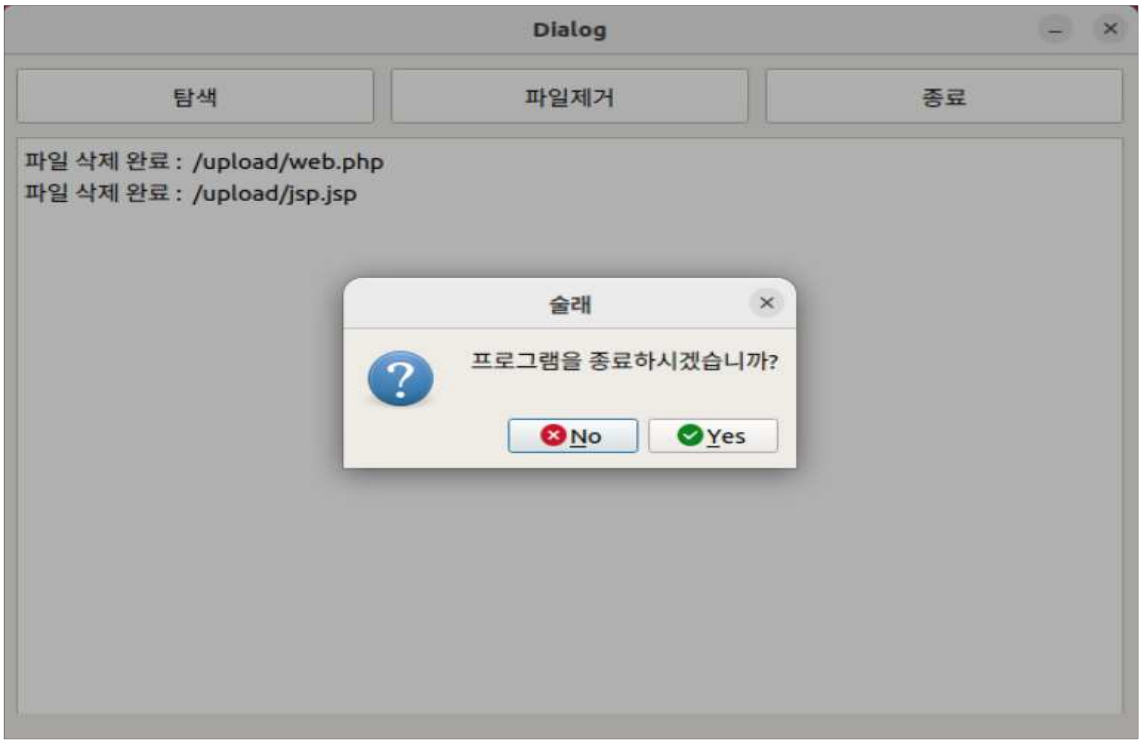
[그림 3-12]. 탐지된 웹쉘 출력

공격에 사용된 웹쉘 파일들을 성공적으로 탐지했다. 해당 상태에서 파일 제거를 누르게 되면 탐지된 웹쉘들은 모두 삭제된다. 파일 제거 버튼을 통한 이벤트를 실행시킨 결과는 [그림 3-13]과 같다.



[그림 3-13]. 탐지된 웹쉘 삭제

종료버튼을 누르게 되면 [그림 3-14]와 화면이 출력되고 프로그램은 종료된다.



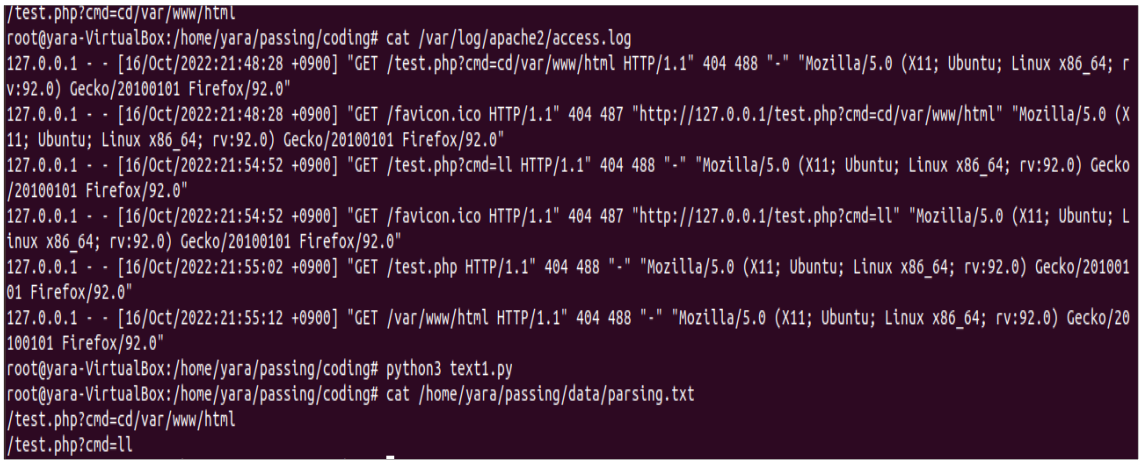
[그림 3-14]. 프로그램 종료

Ⅳ. 분석

본 장에서는 웹쉘 탐지 프로그램이 실행되는 과정을 바탕으로 개선점에 관해 기술한다.

1. Apache Access.log 수집 개선

Apache Access.log는 GET 방식의 로그만 작성한다. 최근 공격에 사용되는 대부분의 웹쉘은 POST 방식을 채택하고 있지만 현재 공격 로그를 파싱하고 있는 Access.log 파일 내에는 GET 방식의 로그만 남게 된다.



[그림 4-1]. 수집된 로그 사진

이러한 문제점을 해결하기 위하여 Apache 환경에서 dumpio 메소드을 이용하여 POST 데이터까지 로그에 남기는 방법을 사용하였다. 하지만 해당 메소드는 작성하는 로그의 크기가 너무 방대하고 POST 방식의 로그를 수집하기 위한 메소드가 아니라고 판단하여 dumpost 모듈을 사용하여 개선을 진행하였다.

2. 탐지 프로세스 개선

현재 웹쉘 탐지 프로그램은 온오프 방식으로 동작한다. 하지만 악성행위는 실시간으로 발생하므로 웹서버나 WAS서버의 웹 소스에서 실시간으로 웹쉘 패턴의 탐지 지원이나 난독화 웹소스의 실시간 웹쉘 패턴의 탐지 지원이 필요하며, 웹 어플리케이션에서 작동되는 웹쉘 패턴에 대한 공격탐지에 대해서도 추가적인 연구가 필요하다.

Ⅴ. 결론

웹이 고도화된 시점에서 웹 보안이 중요시되고 있다. 그중 웹 기반 해킹 공격 중 하나인 웹쉘 공격으로 인한 피해는 증가하는 추세이다. 웹쉘은 Server Side Script로 이루어져 있고 해당 언어들의 특성상 웹쉘을 제작기에 다른 공격 기법에 비해 큰 노력이 필요하지 않다. 따라서 패턴 기반 탐지를 우회하는 웹쉘은 얼마든지 제작되어 진다.

본 논문에서는 이미 침투한 웹쉘을 탐지하는 방안으로 Apache Access log와 Yara Library를 활용한 웹쉘 탐지 프로그램을 제안한다. 제안하는 프로그램을 활용할 경우 탐지를 우회하고 침투한 웹쉘에 대해 Apache Access log를 이용하여 해당 파일의 행위를 감지한다. 이후 의심되는 파일은 Yara Library를 통해 웹쉘 여부를 탐지하고 해당 파일의 유지 혹은 삭제 및 격리를 진행하여 그 동작을 중지시킬 수 있다. 이로 인해 서버 내에서 내포하고 있는 위험도를 한 층 낮출 수 있다고 전망된다. 그러나 해당 프로그램은 간단한 웹쉘을 대상으로는 효율적이지만 점점 고도화되는 웹쉘의 기술에 대응하기에는 기반하는 패턴이 너무 적고 파라미터에 입력되는 웹쉘에 대해서만 탐지할 수 있다는 한계가 명확하다. 따라서 다양한 행위기반 탐지 기법을 적용하는 기술에 관한 지속적인 연구가 필요하다.

참고 문헌

[1] 석진욱, 최문석, 김지명, 박종순. “오픈소스 IDS/IPS Snort와 Suricata의 탐지 성능에 대한 비교 연구” 디지털산업정보학회논문지, 2016

[2] 방재광. “웹쉘 해킹 대응 방안 개선에 관한 연구.“ 국내석사학위논문 동국대학교, 2019

[3] Yixin Wu, Yuqiang Sun, Cheng Huang, Peng Jia, Luping Liu, “Session-Based Webshell Detection Using Machine Learning in Web Logs,“ Security and Communication Networks, 2019.

[4] 김준영, 김규림, 박한솔, 안승환, 이현탁, 최상용, 이종락 “Access_log를 활용한 매크로 프로그램 탐지 시스템 개발, “ 한국컴퓨터정보학회 학술발표논문집, 2020.

[5] 박채금, 노봉남, 김정일. “GET과 POST정보를 이용한 웹 어플리케이션 보안 시스템 설계,” 한국정보과학회 학술발표논문집, 2005

[6] 김창훈, “.YARA 속도 개선을 위한 새로운 S/W 구조 설계,” 한국통신학회논문지, 2016