컴퓨터 보안_01

실습 5주차

동국대학교 CSDC Lab.
실습조교 김선규
2023.04.12 (Wed.)

Index

1. 실습 과제

- 파일 검증 (File Verification)
- 바이러스 서명(Virus Sigature)
- 트로이목마 만들기 (Trojan Horse)
- 악성프로그램 찾기(Finding Malware)
- 루트킷(Rootkit)
- Buffer Overflow(BOF)

2. Q & A

Project 유의사항

➢ Windows 바이러스 및 위협 방지

- 윈도우키 + R (실행) → gpedit.msc → 컴퓨터 구성 → 관리 템플릿 → Windows 구성 요소 → Microsoft Defender 바이러스 백신 → Microsoft Defender 바이러스 백신 끄기 → 사용(E) → 확인
- gpedit.msc 명령어가 불가할 경우, windows home 에디션 때문이며 다음 링크에서 받은 파일을 관리자 권한으로 실행 -> 재부팅 하여 재시도
- https://dguackr-my.sharepoint.com/:u:/g/personal/code_dgu_ac_kr/EV3pFXgKa_NGtPI5ik6z-IsBniqAUPHUp8ppfceIeZMVdw?e=JR9C07
- **해당 컴퓨터 보안 실습을 모두 진행한 후에는 다시 "구성되지 않음(C)"으로 변경하고 윈도우 보안 유지 권장**

파일 검증(File Verification)

➢ 파일 검증(File verification) 이란?

- 파일검증은 특정 알고리즘을 이용하여 파일의 인증성(authencity)과 무결성(integrity)을 확인하는 과정을 말한다.
- 이 중에서 파일 무결성 검사 (file integrity check)란 원본 파일에 대한 변형이나 수정여부를 확인하는 것이다.

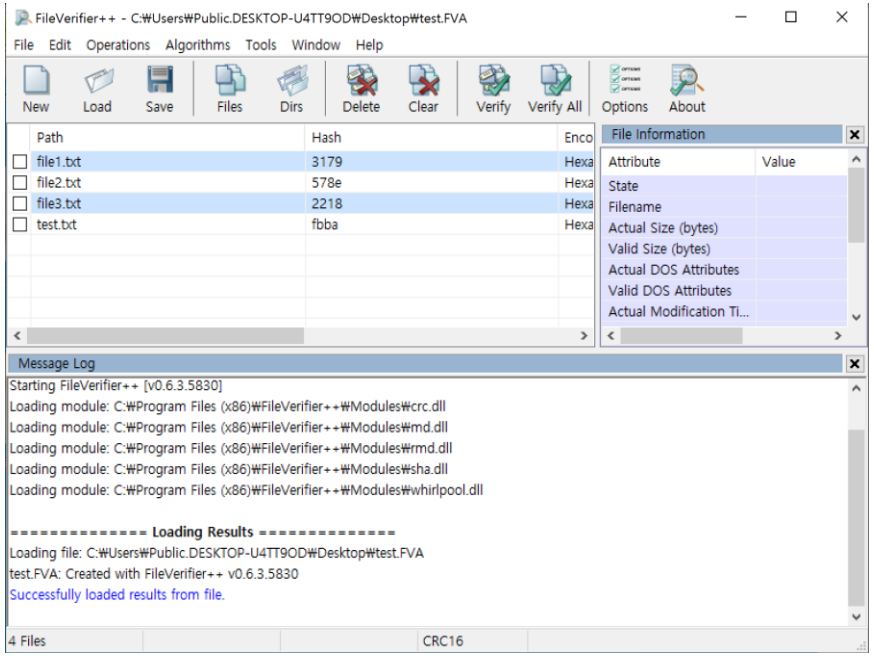
- FileVerifier++ (FV)는 파일의 체크섬을 계산해주는 프로그램으로서 보안 알고리즘을 잘 모르는 사용자라고 하더라도 손쉽게 사용할 수 있다는 장점이 있다.
- 이 프로그램은 파일의 내용이 변형(corruption) 또는 조작(tamper) 되었는지 검증하는데 용이하다.
- FV는 거의 모든 종류의 해쉬 알고리즘 (hash algorithm)들을 지원한다.
- 해쉬 알고리즘은 수학적 함수를 이용하여 사이즈가 크고 다양한 (크기가 일괄적이지 않은) 데이터를 고정된 크기의 작은 데이터로 변환하는 것을 말한다.

파일 검증(File Verification)

dongguk UNIVERSITY

➢ FileVerifier++ (FV)

- 다운로드 링크 [http://www.programmingunlimited.net/siteexec/content.cgi?page=fv]
- [MSI]를 클릭하여 시험중인 버전 말고 Stable 버전을 다운받는다. MSI는 Microsoft Installation의 약자로 윈도우용 설치파일이다. FV는 윈도우 전용 프로그램으로 윈도우에 익숙한 사용자들이 손쉽게 설치할 수 있다.
- [시작] -> [모든 프로그램] -> [FileVerifier++]에서 FileVerifier++를 실행하거나 프로그램이 설치된 폴더에서 fv.exe를 실행시키면 프로그램창이 생성된다. 메뉴와 툴바로부터 그 사용법을 익힌다.



파일 검증(File Verification)

dongguk UNIVERSITY

➢ FileVerifier++ (FV) 실습

1. FV 메뉴의 'Process String'이라는 명령을 이용하여 '감사합니다' 라는 문자열의 CRC32 해쉬값을 구하시오.
2. 3개의 텍스트 파일 (file1.txt, file2.txt, file3.txt)을 준비하고 각각의 해쉬값을 구하시오. 이때 file1.txt와 file2.txt는 CRC16을, file3.txt는 CRC32 알고리즘을 이용하시오.
3. test.txt라는 임의의 파일을 생성하여 해쉬값을 계산하시오 (단, 알고리즘은 어떤 것을 선택하더라도 관계없음).
4. FV의 'Verify' 명령어를 이용하여 위 3번의 test.txt을 검증하되 상태가 'invalid'가 되도록 만들어 보시오.
5. 1~4 번 실습을 통해 알게된 파일 검증 방법에 대해 기술하시오.

바이러스 서명(Virus Signature)

dongguk UNIVERSITY

➢ 바이러스 서명(Virus Signature) 실습

- 본 실습에서는 바이러스 서명을 테스트하는 방법을 경험한다.
- 다음의 테스트 파일은 유럽의 컴퓨터 바이러스 백신 연구소(EICAR)에서 개발한 것으로 안티바이러스 소프트웨어의 기능성 시험을 위해서 사용된다.
- 이번 실습을 수행하기 위해서 윈도우 컴퓨터와 선호하는 안티바이러스 프로그램들이 필요하다.

1. 다음의 정보를 텍스트 파일에 입력하여라.

X5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*

2. 파일 이름을 virusdemo.txt파일로 저장한 다음, 확장자를 실행파일로 변경해서 그 이름이 virusdemo.exe가 되도록 하여라.
3. 안티바이러스 프로그램으로 스캔을 시작하여 virusdemo.exe를 검사한다.
- https://www.virustotal.com/gui/home/upload

바이러스 서명(Virus Signature)

dongguk UNIVERSITY

➢ 바이러스 서명(Virus Signature) 실습

- 사용된 안티바이러스 프로그램은 이 파일을 악성코드로 표시해야만 한다.
- 이 파일은 실제로 바이러스는 아니지만 사용자가 자신의 안티바이러스 소프트웨어(예, 알약, v3)가 제대로 동작하는지 시험하기 위한 방법으로 고안되었다.
- 일부 시스템은 이 파일을 저장하기도 전에 악성코드 차단을 위해 저장을 막을 수도 있다.

트로이 목마 만들기(Trojan Horse)

dongguk UNIVERSITY

➢ 트로이목마 만들기(Trogan Horse)

- 실습에서는 해커가 트로이목마를 배포할 수 있는 방법들 중 한 가지를 경험해본다.
- 기본적으로 오래된 윈도우 시스템은 CD가 CD트레이에 삽입되면 자동으로 CD를 시작한다.
- 이 기술을 이용하여 악성코드 배포를 흉내낼 수 있다. 이 실습을 위해서 공CD와 CD writer가 필요하다.

트로이 목마 만들기(Trojan Horse)

dongguk UNIVERSITY

➢ 트로이목마 만들기(Trogan Horse)

1. autorun.ini라는 이름의 텍스트 파일을 만든다. 파일 안에 다음의 내용을 넣는다.

```
[autorun]
Open paint.exe
Icon=paint.exe
```

2. autorun.ini와 paint.exe를 CD로 구울 대상 폴더 안에 넣는다.

3. CD를 구운후에 CD-ROM 드라이브에 넣고 결과를 관찰한다. CD는 자동으로 시작되어 paint.exe 프로그램이 실행되어야 한다.

4. 결과에 관해 생각해보자. 이 실습은 특별한게 없어 보이지만, 트로이목마 프로그램을 정상적인 프로그램으로 쉽게 포장해서 사용했다. CD를 근처에 그냥 놔두거나 '2020년 보너스'와 같이 이목을 끄는 제목을 붙여놓으면 누군가가 가져가서 내용을 보도록 할 수 있다. 누군가가 CD를 실행한다면 감염이 되고 만다. 자동실행 기능을 꺼놓았더라도 사용자가 CD-ROM아이콘을 더블클릭하면 여전히 프로그램은 실행된다.

➢ 악성프로그램 찾기(Finding Malware)

- 이번 실습에서는 컴퓨터 시스템에 존재하는 악성코드 또는 데이터 탈취 도구를 찾는 일반적인 방법을 살펴본다.

1. 자신의 컴퓨터에 아직 트로이목마가 설치되어 있지 않다면(좋은 방법은 아니지만) 찾아야할 것이 있다. http://netcat.sourceforge.net/download.php로 가서 윈도우용 넷캣(Netcat)을 다운로드 한다.
2. 다음과 같은 명령을 사용하여 자신의 컴퓨터에 설치된 넷캣 리스너를 시작한다. : nc -n -v -l -p 12345
3. 넷캣이 시작되어 리스닝 상태에 있게 되면 마우스 오른쪽 버튼으로 작업표시줄을 클릭하여 작업관리자를 실행한다. 어플리케이션에 넷캣이 실행중인 것이 명확하게 보여야한다.
4. 새로운 명령 프롬프트를 열고 netstat -an을 입력한다. 다음과 유사한 리스닝 목록이 보여야 한다.

```
C:₩>netstat –an
Active Connections
Proto      Local Address          Foreign Address        State
TCP        0.0.0.0:80             0.0.0.0:0              LISTENING
TCP        0.0.0.0:445            0.0.0.0:0              LISTENING
TCP        0.0.0.0:1025           0.0.0.0:0              LISTENING
TCP        0.0.0.0:1027           0.0.0.0:0              LISTENING
TCP        0.0.0.0:12345          0.0.0.0:0              LISTENING
```

[Fig 3. netstat –an 결과]

➢ 악성프로그램 찾기(Finding Malware)

- 결과에는 80번 포트가 리스닝 상태여야 한다. 자신의 컴퓨터 리스닝 목록에서 일반적이지 않은 무엇인가를 인지했는가? 여기서 보이는 목록에서 뭔가 이상한 점이 확인되는가? 마지막 줄에서 넷버스(NetBus)의 기본 포트인 12345번 서비스 포트가 리스닝임을 보여준다.

5. 브라우저를 사용해 http://technet.microsoft.com/en-us/sysinternals/bb897437.aspx로 가서 TCPView를 다운로드하자. 이것은 무료 버전의 GUI기반 프로세스 뷰어로, 실행중인 프로세스에 관해 netstat보다 아주 자세한 정보를 보여준다. 로컬과 원격 주소, TCP커넥션 상태를 포함해서 시스템의 모든 TCP와 UDP 말단 정보를 제공한다. 넷캣이 여전히 실행중이라면 쉽게 찾을 수 있어야 한다.

6. TCPView를 종료하고 https://technet.microsoft.com/en-us/library/bb897437.aspx로 간다. 이곳에서 프로세스 뷰어(process viewer)라는 또 다른 프로그램을 다운로드할 수 있다. TCPView와 유사함을 알 수 있을 것이다.

[Fig 3. netstat –an 결과]

➢ 악성프로그램 찾기(Finding Malware)

7. 마지막으로 무소프트의 클리너(MooSoft's Cleaner)라는 트로이목마 제거 도구를 살펴보자. 이 도구는 트로이목마, 웜, 키로거, 스파이웨어로부터 컴퓨터와 데이터를 안전하게 지키도록 설계된 프로그램 시스템이다. http://en.kioskea.net/download/download-16047-moosoft-s-the-cleaner에서 다운로드할 수 있다. 설치후에 프로그램을 실행하여 넷캣이나 다른 프로그램을 찾아내는지 확인해보자.

〈실습을 마친 후, 본 실습을 수행하는 동안 설치된 넷캣이나 기타 프로그램을 제거하는 것이 좋다〉

루트킷(Rootkit)

➢ 루트킷(Rootkit) 이란?

- 루트킷 (rootkit)은 컴퓨터 소프트웨어 중에서 악의적인 것들의 모음으로써, 자신의 또는 다른 소프트웨어의 존재를 가림과 동시에 허가되지 않은 컴퓨터나 소프트웨어의 영역에 접근할 수 있게 하는 용도로 설계되었다.

➢ 루트킷(Rootkit) 실습 - in Linux

- 이번 실습은 루트킷 검사기를 다운로드하여 설치하고 다양한 옵션을 확하는 것이다.
- 루트킷 헌터(Rootkit Hunter)는 오픈소스로 리눅스 기반 시스템에서 동작하는 루트킷 및 유사한 도구의 존재 여부를 검사한다.
- 루트킷 헌터는 (http://rkhunter.sourceforge.net/)에서 다운로드할 수 있다. 이 프로그램은 리눅스 시스템에 다운로드하여 설치할 수 있으며, 와일리 웹사이트 (wiley.com/go/networksecuritytestlab) 에서 찾을 수 있는 칼리 리눅스에도 포함되어 있다.

➢ 루트킷(Rootkit) 실습 - in Linux

1. 일단 리눅스 시스템을 시작하고, 루트 터미널을 열어 루트킷헌터를 다운로드한다. 명령중에 다음의 명령을 입력한다.

```
wget http://downloads.rootkit.nl/rkhunter-<version>.tar.gz
```

<version>구문에는 소프트웨어의 현재 버전을 입력한다.

2. 다운로드가 완료되면 다음과 같은 명령을 입력하여 압축을 해제한다. 이 명령은 루트킷헌터의 압축을 풀어낸다.

```
tar zxf rkhunter-<version>.tar.gz
```

3. 루트킷헌터의 설치를 위해 루트킷헌터 폴더로 이동한다.

```
cd rkhunter
```

4. 적절한 디렉토리로 이동한 후에, 스크립트를 실행하여 설치를 완료한다. 이를 위해 다음과 같이 입력한다.

```
./installer.sh
```

➢ 루트킷(Rootkit) 실습 - in Linux

5. 루트킷헌터가 설치되면 프로그램을 실행한다. 사용할 수 있는 다양한 옵션이 있다. 완전한 시스템 점검 수행을 위해 다음 명령을 실행한다.

Rkhunter-checkall

루트킷헌터는 많은 종류의 루트킷을 검색할 수 있는데 그중 일부는 다음과 같다.

55808 Trojan - Variant A

ADM W0rm

AjaKit

aPa Kit

Apache Worm

Ambient (ark) Rootkit

…

스캔이 완료되면 다음과 유사한 메시지를 받는다.

➢ 루트킷(Rootkit) 실습 - in Linux

```
---------------------------- Scan results ----------------------------
MD5
MD5 compared: 0
Incorrect MD5 checksums: 0
File scan
Scanned files: 399
Possible infected files: 0
Application scan
Vulnerable applications: 9
Scanning took 15748 seconds
----------------------------------------------------------------------
Do you have some problems, undetected rootkits, false positives, ideas
or suggestions?
Please email me by filling in the contact form (@http://www.rootkit.nl)
----------------------------------------------------------------------
```

[Fig 3. 스캔 결과 레포트]

- 본 실습에서는 시스템이 감염되지 않은 것으로 확인된다. 하지만 감염이 되었다면 많은 도전과제에 직면하게 될 것이다. 루트킷을 깨끗이 청소하기는 거의 불가능하기 때문이다. 은닉하기는 루트킷의 주요 목적이기 때문에 감염된 모든 잔재가 삭제되었다고 단언하기는 어렵다. 그래서 항상 정상적인 매체를 이용해 재설치해야만 한다.

➢ Buffer overflow 취약점 실습

- 이 실습의 목적은 학생들이 강의를 통해서 공부한 버퍼 오버플로우 취약점에 대한 내용을 실제로 해봄으로써 기초적인 경험을 할 수 있도록 하는 것이다. 버퍼 오버플로우는 어떤 프로그램이 미리 할당된 고정된 크기의 버퍼 경계를 넘어서 데이터를 쓰려고 시도하는 조건으로써 정의된다. 이러한 취약점은 코드의 임의의 부분을 실행하는 것만으로도 악의적인 사용자에 의해 프로그램의 흐름 제어를 변경하기 위해서 이용될 수 있다. 이러한 취약점은 버퍼 같은 데이터 저장소와 리턴 주소같은 제어의 저장소가 혼재되어 있기 때문에 발생한다. 즉, 데이터 저장소 부분의 오버플로우가 리턴 주소를 변경할 수 있기 때문에 프로그램의 제어 흐름에 영향을 주게 된다.

- 이 실습에서 학생들은 버퍼 오버플로우 취약점을 가진 프로그램을 사용하게 될 것이다. 학생들이 해야할 일은 이 취약점을 이용해서 루트 권한을 얻을 수 있는 기법을 개발하는 것이다. 루트 권한 탈취 공격이외에도 우리는 오버플로우 공격에 대항하기 위해서 리눅스(페도라)에서 이미 구현 되어있는 여러 가지 보호 기법을 살펴볼 것이다. 우리는 이러한 기법들이 제대로 동작하는지 평가해볼 것이다.

➢ Buffer overflow 취약점 실습 주의사항

- 이 실습의 결과는 운영체제에 따라서 다르게 나온다는 것을 주의해야 한다. 본 실습에 대한 설명과 논의는 리눅스 페도라(코어 4 또는 5)에 기반을 두고 있다. 최신 페도라 리눅스 버전에서도 또한 동작 해야한다. 그러나 다른 운영체제를 사용한다면 다른 문제들과 이슈들이 나타날 것이다.

➢ 과제 내용

- 별도의 파일에서 실습에 대한 자세한 내용과 과제를 설명한다. 여러분이 직접 해보고 경험한 것들을 실습 레포트에 자세히 반영하여 제출하시오. 또한 실습을 하는 동안 개인적으로 흥미가 있었거나 놀라운 경험에 대한 설명을 포함하는 것이 좋다.

➢ 필요한 파일

- stack.c (취약한 프로그램) / call_shellcode.c / exploit.c.

Buffer overflow [추가 자료]

dongguk UNIVERSITY

➢ Buffer overflow(BOF) 보호 해제

- Fedora를 포함한 여러 Linux 운영체제는 BOF로부터 보호하기 위한 보호 기법들이 적용되고 있으며, 이를 해제해야함.

➢ 주소 공간 레이아웃 무작위화 해제

- 주소 공간 레이아웃 무작위화(ASLR, Address Space Layout Randomization)는 메모리 오염 취약점의 악용을 방지하는 보안 기술이며, 이는 스택, 힙, 그리고 라이브러리의 주소 공간에서의 위치를 무작위로 배열하여, 안정적으로 악의적인 코드로 점프하는 것을 방지한다. 이는 다음과 같은 명령어로 해제할 수 있다.
- sysctl -w kernel.randomize_va_space=0 (Fedora 4에서 sysctl은 /sbin/sysctl 에 존재함.)

➢ ExecShield 해제

- ExecShield는 레드햇에 의해 시작된 프로젝트로, 버퍼 오버플로와 같은 메모리 관련 취약점에 대한 패치이다. 주로 코드 세그먼트 제한을 활용하며 셸코드의 삽입 및 실행의 난이도를 증가시킨다. 이는 다음과 같은 명령어로 해제할 수 있다.
- sysctl -w kernel.exec-shield=0

Buffer overflow [추가 자료]

dongguk UNIVERSITY

➢ 기본 셸 zsh로 변경하기

- zsh 설치를 위해 /etc/yum.repo.d/에 있는 세 파일의 baseurl을 다음과 같이 수정

fedora-extras.repo
baseurl=http://archive.fedoraproject.org/pub/archive/fedora/linux/extras/$releasever/$basearch/

fedora-updates.repo
baseurl=http://archive.fedoraproject.org/pub/archive/fedora/linux/core/updates/$releasever/$basearch/

fedora.repo
baseurl=http://archive.fedoraproject.org/pub/archive/fedora/linux/core/$releasever/$basearch/os/

- zsh 설치 명령어
  - yum install zsh
- 이후 기존 심볼릭 링크 sh 삭제 및 zsh에 대한 새로운 심볼릭 링크 sh 생성
  - cd /bin/
  - rm sh
  - ln -s zsh sh
  - ls -l sh

Buffer overflow [추가 자료]

dongguk UNIVERSITY

➢ 스택 가드 해제

- 실습 교안에서는 gcc로 컴파일 시 스택 가드 해제를 위해 -fno-stack-protector 옵션을 사용하라고 함.
- 스택가드 옵션은 gcc 4.1 버전부터 지원되며, Fedora 4에서는 gcc 4.0 버전을 사용하기 때문에 해당 옵션을 사용하지 않아도 됨.

Q & A