침해사고 분석대응 전문가

리눅스 침해사고 분석

ADT캡스|인포섹

Top-CERT
강사 : 이 화 목 책임
(leehwamock@sk.com)

목 차

1. Introduction

- Threat, Risk, Vulnerability

2. Getting Started

- Basic Considerations
- Viewpoint
- Case Study

3. Linux 침해사고 분석

- Overview
- Collecting Artifacts
- Artifacts Analysis
- Memory Analysis
- Linux File Recovery

1. Introduction – Threat, Risk, Vulnerability

□ Threat(위협)

- ▶ a potential cause of an incident that may result in harm to a system or organization
- ▶ 시스템이나 조직에 피해(손해)를 가할 수 있는 사고의 잠재적 원인

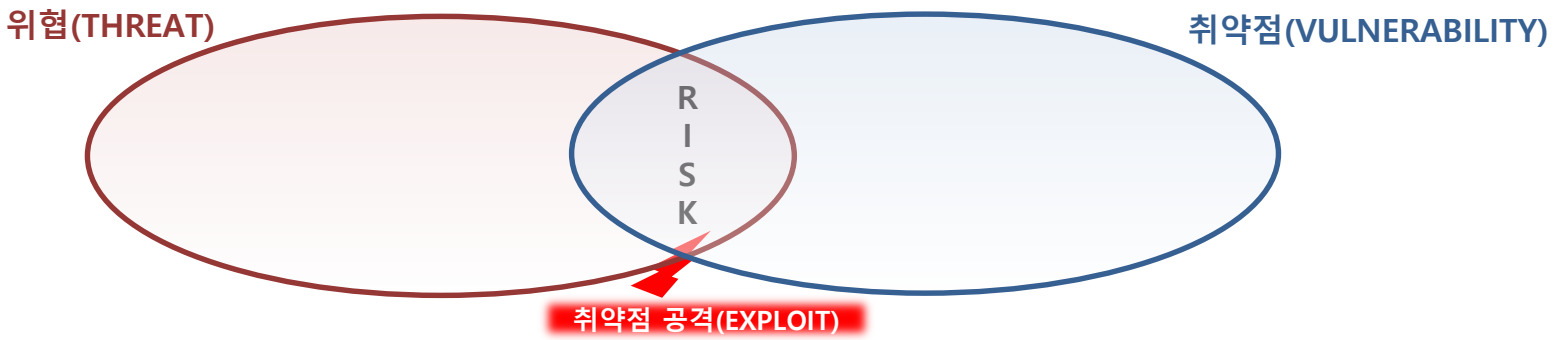
□ Vulnerability(취약점)

- ▶ a weakness of an asset (resource) or a group of assets that can be exploited by one or more threats
- ▶ 하나 이상의 위협에 의해 악용될 수 있는 자산(자원)의 또는 자산 그룹의 약점

□ Risk(위험)

- ▶ potential for loss, damage, or destruction of an asset as a result of a threat exploiting a vulnerability
- ▶ 취약점을 이용한 위협의 결과로 인한 자산의 손실, 손상, 파괴의 가능성

1. Introduction – Threat, Risk, Vulnerability (Cont.)



Risk
- business disruption
- financial losses
- loss of privacy
- damage to reputation
- loss of confidence
- legal penalties
- impaired growth
- loss of life

=

Threats
- angry employees
- dishonest employees
- criminals
- governments
- terrorists
- the press
- competitors
- hackers
- nature

X

Vulnerabilities
- software bugs
- broken processes
- ineffective controls
- hardware flaws
- business change
- legacy systems
- Inadequate BCP
- human error

Information Security Risks, Threats and Vulnerabilities
© simplicable.com

[참고/출처]
https://arch.simplicable.com/arch/new/the-big-list-of-information-security-vulnerabilities

목 차

1. Introduction
   - Threat, Risk, Vulnerability
2. **Getting Started**
   - Basic Considerations
   - Viewpoint
   - Case Study
3. Linux 침해사고 분석
   - Overview
   - Collecting Artifacts
   - Artifacts Analysis
   - Memory Analysis
   - Linux File Recovery

2. Getting Started – Basic Considerations

☐ Prerequisite Knowledge and Skills

2. Getting Started – Basic Considerations (Cont.)

□ Download Tools

▶ **Windows Sysinternals**
https://docs.microsoft.com/en-us/sysinternals/

▶ **NirSoft**
http://www.nirsoft.net/

▶ **Digital Forensic Research Center(고려대학교)**
http://forensic.korea.ac.kr/

2. Getting Started – Basic Considerations (Cont.)

□ Interview

2. Getting Started – Basic Considerations (Cont.)

□ Backup

2. Getting Started – Basic Considerations (Cont.)

□ 자료 수집

항목	설명
사고 이력 확인	최초 사고 징후 확인 날짜 및 시간
	특이사항 및 조치사항 확인
서비스 개요 및 현황 확인	서비스 개요 정보 수집 - 서비스 관리조직/담당, 서비스개요, 주요자산 등
	관리적 보호수준 확인 - 사고대응절차, 자산관리, 접근통제, 암호화 등
환경설정 확인	로그 위치 및 홈 디렉토리 확인
	새로 추가된 가상 사이트 및 디렉토리 확인
	방화벽 정책 설정 확인
웹구조 파악	운영되고 있는 사이트들 확인
	네트워크/웹 방화벽 적용여부 확인
	네트워크 전체 구조 확인(구조도)
	운영 중인 서버 및 역할 확인 (DB서버 등)
	DB연결, 공유 연결 등 확인

2. Getting Started – Basic Considerations (Cont.)

□ 자료 수집 (Cont.)

항목	설명
로그수집	WEB로그(accesslog, errorlog)
	WAS로그(accesslog, errorlog)
	OS로그
	보안장비(FW, IDS, IPS, WAF, etc.) 로그
DB 점검	DB사용자 확인(관리자 확인 필요) - 엔터프라이즈 관리자, SA 계정 패스워드 확인
	최근 생성된 DB테이블 확인 - SQL Injection 도구 사용 시그니쳐 : D99_Tmp 등
	안정적인 버전 사용여부
웹쉘 점검	웹쉘 시그니쳐 점검(정규표현식 포함), 웹쉘 파일명으로 검색
시스템 정보 확인	운영체제
	어플리케이션
	프레임워크
	시간

2. Getting Started – Basic Considerations (Cont.)

□ TimeLine

▶ 침해사고분석 측면에서의 효용

- 분석할 로그의 양을 한정시킴과 동시에 침해 관련 이벤트를 발견할 확률을 높임
  - √ 특정 침해 이벤트에서 획득한 시간 정보를 기준으로 시간 값이 있는 다른 로그를 분석
  - √ 시스템 level의 로그 뿐만 아니라 보안장비에서 발생한 이벤트 등
    시간 값을 갖고 있는 모든 로그를 대상으로 함
- 침해 시나리오 재구성에 따른 논리적 근거 획득
  - √ 침해 관련 시점을 확정 지어 이와 관련된 모든 이벤트를 시간순으로 재구성
  - √ 침해 흐름을 직관적으로 이해하기 위한 근거가 됨

2. Getting Started – Basic Considerations (Cont.)

□ TimeLine (Cont.)

▶ 적용 사례

- 침해 시점, 특히 취약점을 이용한 최초 침해 시점의 이벤트는 시스템 권한을 획득하거나 백도어를 설치하기까지 제한적인 시간동안 이뤄짐
- 분석 기준 시점 획득의 예
  - √ 침해와 관련된 증상이 최초로 발생한 시간
  - √ 위와 관련하여 보안장비 이벤트가 탐지된 시간
  - √ 시스템에서 발견된 악성코드의 파일 생성 시간
  - √ 백신 등에서 다수의 악성코드가 탐지된 시간
  - √ 확인되지 않은 시스템 계정이 생성된 시간, 첫 로그온 시간

2. Getting Started – Basic Considerations (Cont.)

□ TimeLine (Cont.)

▶ 시간 값을 갖고 있는 데이터의 종류

- 파일의 MAC time
- 웹 로그
- 시스템 로그
- 윈도우 이벤트 로그
- 윈도우 레지스트리
  - √ 계정별 최종 로그인 시간, 최근 실행된 파일 내역, network drive 연결 시간, USB 연결 시간 등
  - √ 특정 레지스트리 키 값의 마지막 수정 시간
- 인터넷 히스토리
- 프리패치
- AV 이벤트 로그
- 기타(보안 장비 로그 등)

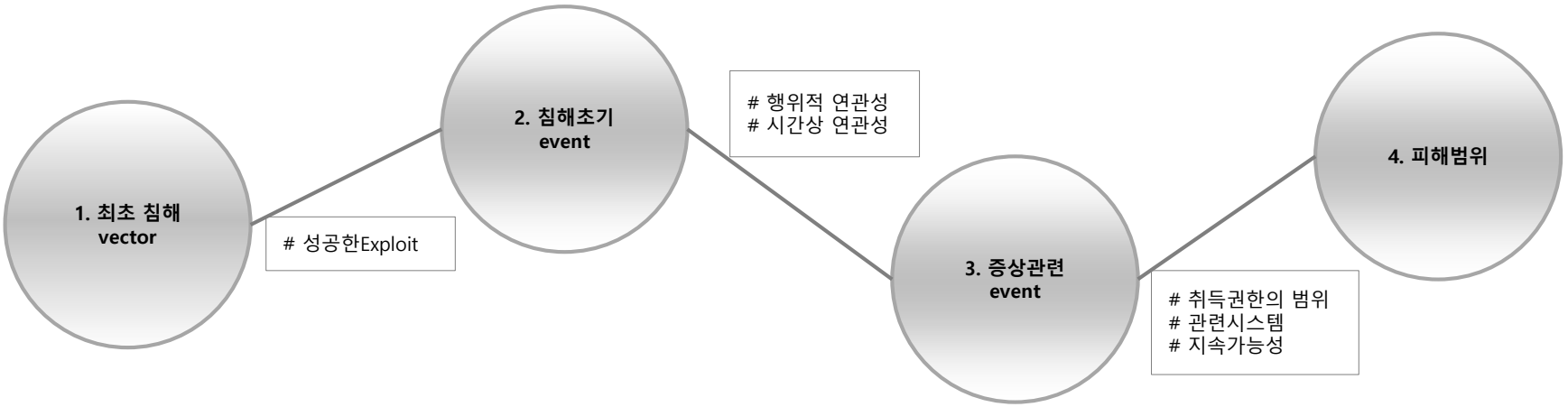
2. Getting Started – Viewpoint

시스템에 **어떻게** 들어왔는가

들어와서 **어떤 행위**를 했는가

무엇으로 어떤 증상을 유발했는가

추가적인 **피해 가능성**은 없는가

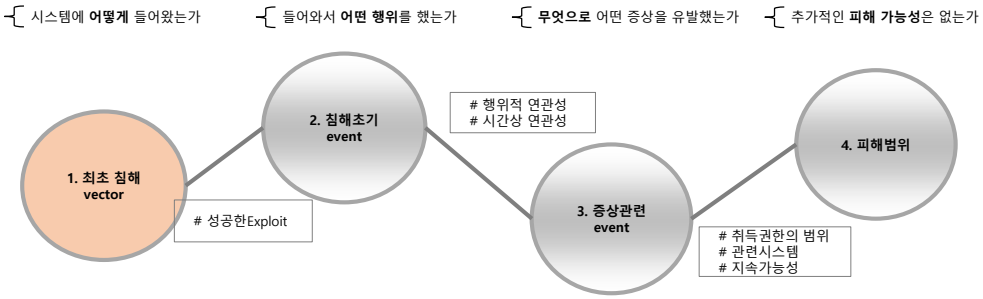


2. Getting Started – Viewpoint (Cont.)

□ 시스템에 어떻게 들어왔는가?

▶ Viewpoint
- 내부 관점에서 볼 것인가 외부 관점에서 볼 것인가
- 내부-외부 관점, 내부-내부 관점에서 어떤 취약 포인트들이 있는가
- 최근 발생한 보안 위협(보안 이벤트)과 취약 포인트와의 교집합엔 무엇이 있나
- 침해 관련 증상이 처음 발생한 시점 이전 수일간 발생한 이벤트/로그 중 특이사항이 있는가
- 해당 서버에서의 사용자 행위가 존재하는가
- 이전에 침해사고를 당한 적이 있는가, 있었다면 당시에 발견된 취약점은 언제 패치 되었는가

▶ Example
- 웹 (어플리케이션) 취약점
- 시스템 취약점
- 사회공학 기법

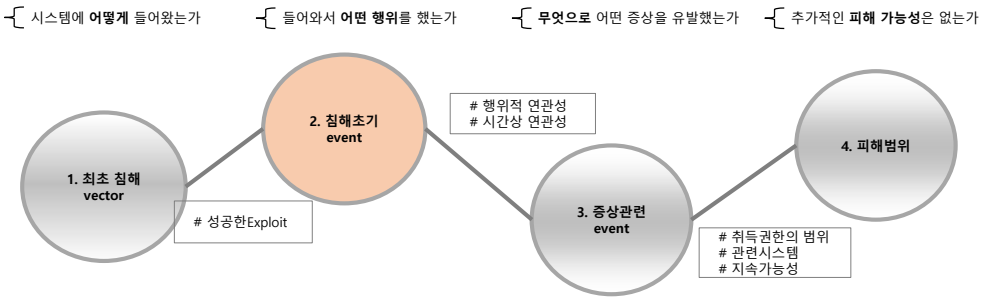


2. Getting Started – Viewpoint (Cont.)

□ 들어와서 어떤 행위를 했는가?

▶ **Viewpoint**
- 최초 침해 event 전후로 시스템 로그상 특이사항이 있는가
- 위 시점 전후로 파일 개체가 새로 생성되거나 변경된 이력이 있는가
- 확인되지 않은 시스템 계정이 생성되어 있다면, 해당 계정으로 무슨 행위를 했는가
- 시스템 로그 등이 변조되거나 삭제된 흔적이 있는가
- 공격자가 키로거나 백도어를 설치했는가

▶ **Example**
- 시스템 계정 생성
- 백도어, 루트킷 설치
- 시스템 파일 변조
- 악성코드 재실행 수단 확보
- 로그 훼손 및 삭제, Anti-Forensic



2. Getting Started – Viewpoint (Cont.)

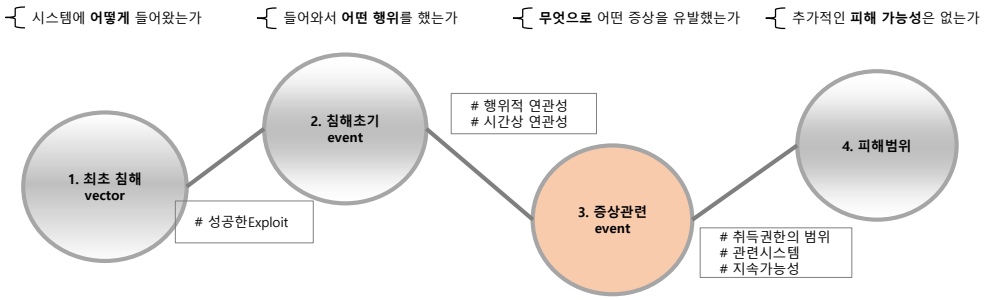
□ 무엇으로 어떤 증상을 유발했는가?

▶ Viewpoint

- 시스템에서 악성코드가 발견되었는가
- 발견된 악성코드의 기능은 무엇인가, 악성코드로 무엇을 할 수 있는가
- 위 악성코드가 실행된 흔적이 있는가
- 위에서 확인된 내용과 침해 증상이 직간접적으로 관련이 있는가
- 악성코드가 아닌 일반 툴이나 내부명령어 등으로 동일한 증상을 재현할 수 있는가
- 증상이 최초로 발생한 시점은 언제인가
- 동일한 증상이 다수 발생했다면 매 시기마다 공통적으로 발견되는 특징이 있는가, 그 특징은 어디서 유래하는가

▶ Example

- DB (개인/기밀정보) 유출
- 과도한 트래픽 발생, DDOS 공격 등으로 인한 서비스 지연/중단
- 페이지 위/변조, 악성코드 삽입
- 기타 침해 이벤트

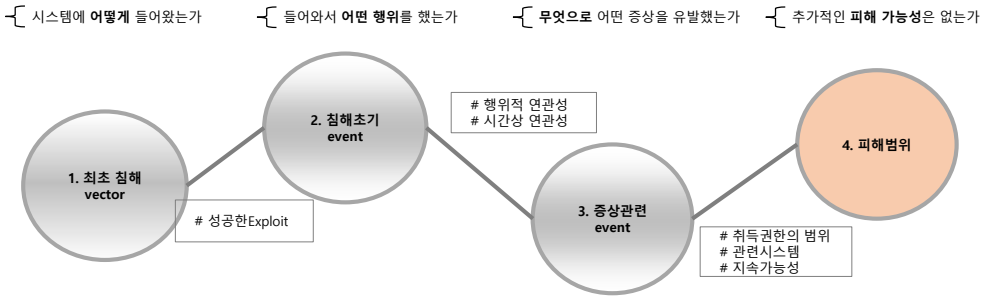


2. Getting Started – Viewpoint (Cont.)

□ 추가적인 피해 가능성은 없는가?

▶ **Viewpoint**
- 네트워크 구성상 조사 대상 서버와 인접한 시스템에는 어떤 것들이 있는가
- 침해 증상이 발생한 시점 전후로 관련 시스템에서 발견된 특이 이벤트/로그가 있는가
- 악성코드 이슈라면 관련 시스템에 동일한 악성코드가 설치 되어 있지는 않은가
- 관련 시스템에서 동일한 증상이 발생한 적이 있는가
- 조사 대상 서버와 동일한 취약점을 가진 시스템이 있는가
- 조사 대상 서버가 해킹 경유지로 사용된 정황은 없는가

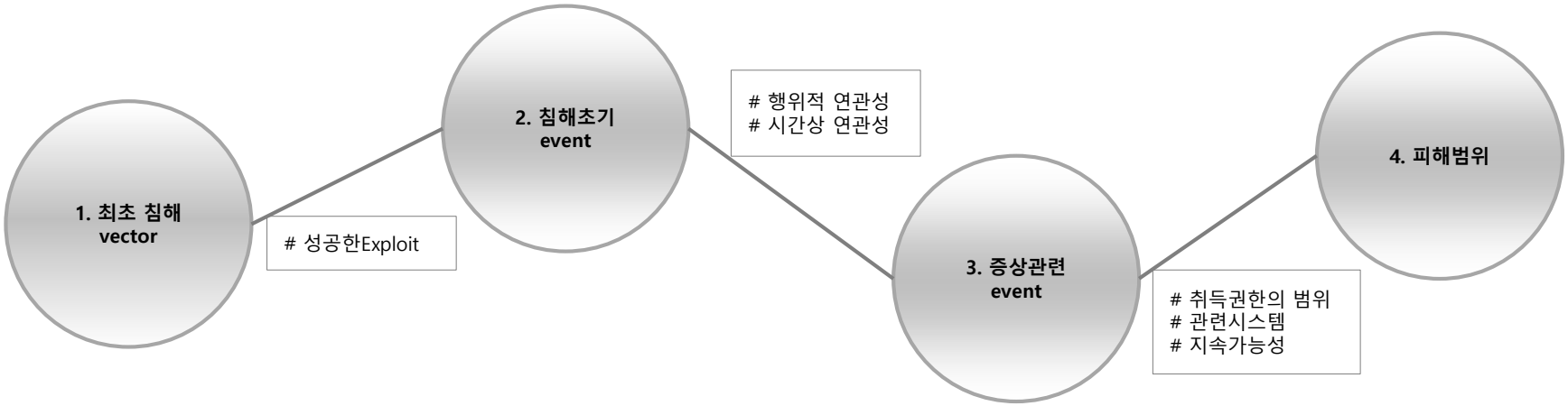
▶ **Example**
- 피해 시스템 경유지로 악용
- 내/외부 시스템 공격지로 악용



2. Getting Started – Case Study

□ 사고분석사례 A

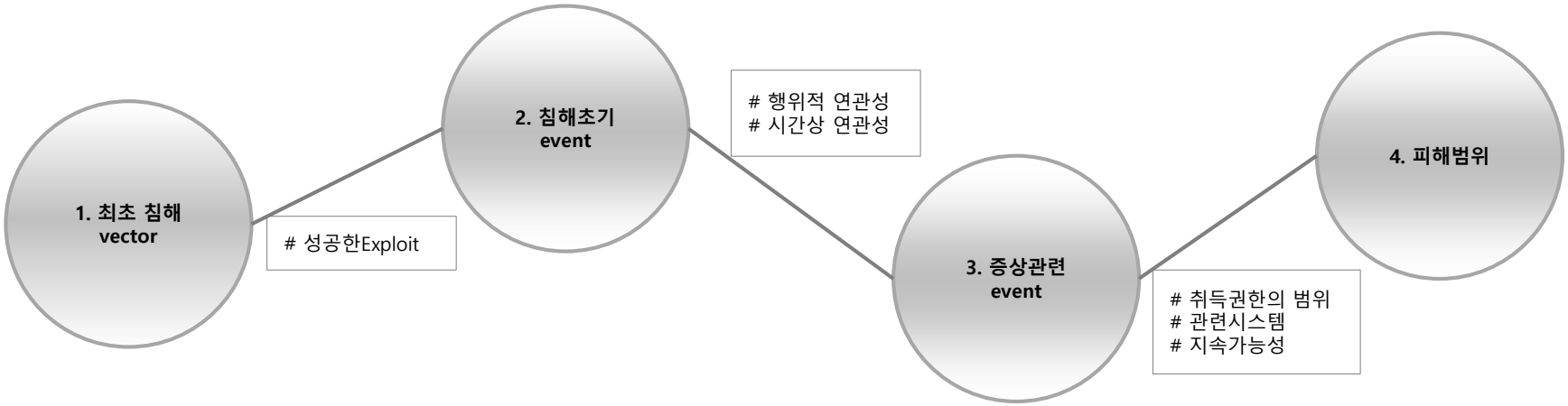
시스템에 **어떻게** 들어왔는가	들어와서 **어떤 행위**를 했는가	**무엇으로** 어떤 증상을 유발했는가	추가적인 **피해 가능성**은 없는가
• Apache-tomcat 관리자 페이지 • war 형태로 웹쉘 업로드	• 시스템 계정 생성 • 시스템 백도어 설치 • 키로거 등 추가 악성코드 설치	• Syn flood tool로 다수의 트래픽 발생 • 위 트래픽으로 인한 네트워크 장애 발생 > 웹서비스 가용성 침해	• 공유폴더가 설정된 타 시스템에 웹쉘 업로드



2. Getting Started – Case Study (Cont.)

□ 사고분석사례 B

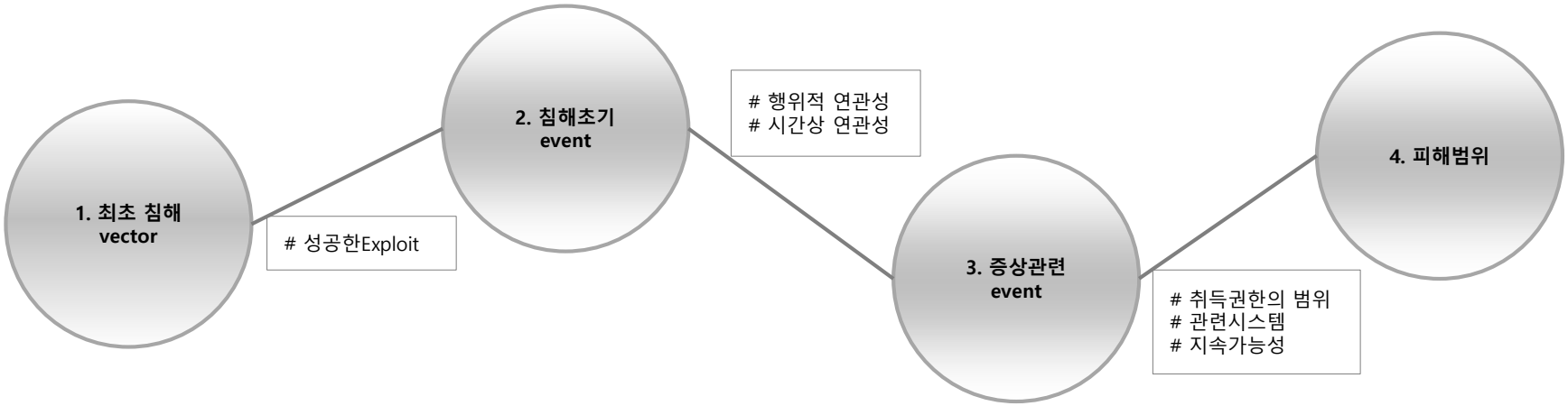
시스템에 **어떻게** 들어왔는가	들어와서 **어떤 행위**를 했는가	**무엇으로** 어떤 증상을 유발했는가	추가적인 **피해 가능성**은 없는가
• PC 자산관리 솔루션 Default 계정 사용 • Client Program을 통해 외부에서 접속	• 악성코드 설치 • DB 서버 접근	• 악성코드를 사용한 서버 정보 유출 • 정상 DB Client를 사용하여 Database 접속 및 개인정보 약 42만건, 기업 내부정보 유출	• 내부 시스템에 악성코드 전파 • 악성코드 유포지로 사용



2. Getting Started – Case Study (Cont.)

□ 사고분석사례 C

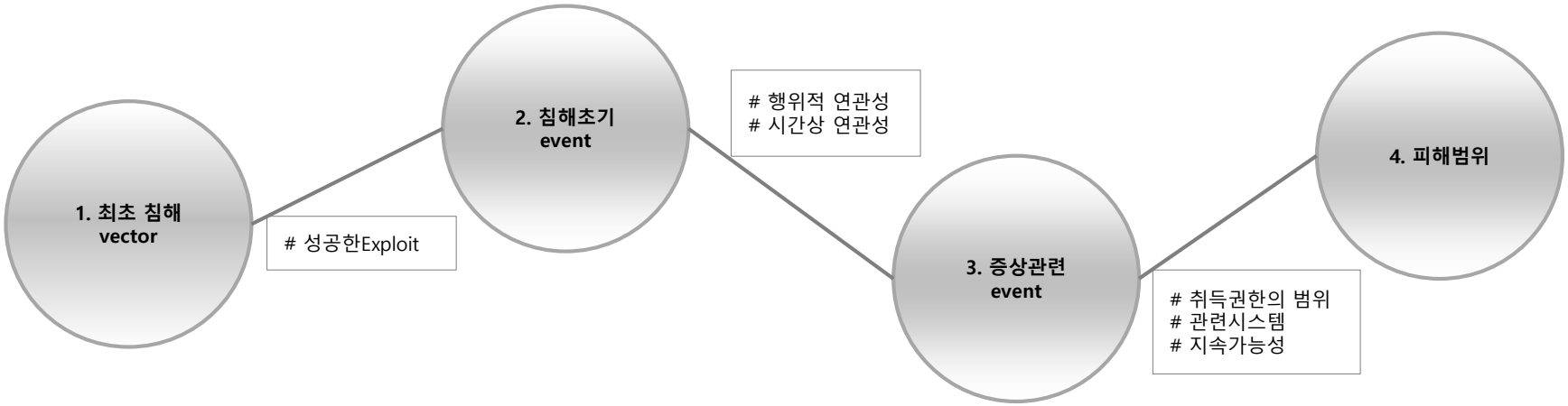
시스템에 **어떻게** 들어왔는가	들어와서 **어떤 행위**를 했는가	**무엇으로** 어떤 증상을 유발했는가	추가적인 **피해 가능성**은 없는가
• 원격데스크톱 Brute-forcing • 작업을 목적으로 취약한 임시 암호로 변경한 시점에 공격 성공	• 시스템 백도어 설치 • 시스템 계정 생성 • VMware 및 OS, BF Tool 설치 • 생성 계정 및 이벤트 로그 삭제	• AntiVirus가 정상적으로 동작하지 않음 • 관리자가 설치하지 않은 임의의 프로그램(VMware)이 확인됨	• 외부 임의 IP로 RD Brute-forcing • 모두 실패한 것으로 확인



2. Getting Started – Case Study (Cont.)

□ 사고분석사례 D

시스템에 **어떻게** 들어왔는가	들어와서 **어떤 행위**를 했는가	**무엇으로** 어떤 증상을 유발했는가	추가적인 **피해 가능성**은 없는가
• 웹 어플리케이션 취약점 (struts) • 웹 소스 코드 취약점 • 웹쉘 업로드	• 업로드 한 웹쉘 1000회 이상 호출	• 웹쉘을 통하여 주기적 DB 유출 • 유출된 DB 정보가 사용된 정황	• 특이사항 없음



목 차

1. Introduction
   - Threat, Risk, Vulnerability
2. Getting Started
   - Basic Considerations
   - Viewpoint
   - Case Study
3. **Linux 침해사고 분석**
   - Overview
   - Collecting Artifacts
   - Artifacts Analysis
   - Memory Analysis
   - Linux File Recovery

3. Linux 침해사고분석 – Overview

☐ Linux Directory

Directory		용도
/	Root	모든 파일과 디렉토리는 "/" 에서 시작, root user 만이 쓰기 권한을 가짐
/bin	User binaries	모든 사용자가 사용하는 명령어(ex ; ps, ls, ping, grep 등) 저장
/sbin	System Binaries	관리자가 시스템 유지를 위해 사용하는 명령어 (ex ; iptables, reboot, ifconfig 등) 저장
/etc	Configuration Files	모든 프로그램에서 필요로 하는 설정 파일
/dev	Device Files	터미널 디바이스, USB 등 시스템에 붙이는 장치 파일들을 저장
/proc	Process Information	프로세스, 시스템 자원 등을 가상의 파일시스템으로 저장(pid, uptime 등)
/var	Variable Files	각종 로그 파일 , 메일 임시 저장
/tmp	Temporary Files	임시 파일 공간
/usr	User Programs	프로그램 설치 공간으로 관련 명령어 , 라이브러리 위치
/home	Home Directories	시스템 계정 사용자들의 홈 디렉토리
/boot	Boot Loader Files	커널을 포함하는 부팅 시 필요한 파일 저장 위치
/lib	System Libraries	라이브러리가 저장 되는 공간으로 대부분 공유 라이브러리 위치
/opt	Optional add-on Applications	개별 판매자의 add-on 어플리케이션을 포함
/mnt	Mount Directory	외부 장치 mount 공간

3. Linux 침해사고분석 – Overview (Cont.)

□ Windows / Linux 차이점

▶ Different file system(다른 파일 시스템)

- 파일 삭제 시 주요 메타데이터가 와이핑(ext3, ext4)
- 파일의 생성시간이 존재하지 않음(ext4 이전)
- LVM 등 논리적인 볼륨을 구성 가능

√ LVM : 여러 개의 하드디스크를 합쳐서 한 개의 파일시스템으로 사용하거나 합쳐진 것을 다시 여러 개로 나누어 파일 시스템 구성이 가능하게 하는 기술

▶ No Registry(레지스트리가 존재하지 않음)

- 운영체제 / 어플리케이션의 실시간 정보 관리를 위한 특별한 구조가 미존재
- 시스템 정보를 산재된 자료로부터 수집하여야 함

▶ Files/data are mostly plain text(텍스트 기반의 파일 데이터)

- 자체 내장 명령어를 조합하여 문자열 검색/필터 만으로 많은 작업 수행 가능

3. Linux 침해사고분석 – Overview (Cont.)

☐ Overview

범주	세부 범주	개요
증거수집	memory dump	시스템 메모리 덤프 수집
	Physical dump	저장 매체 덤프 수집 직접, 원격, Dump file 활용 형태로 수집
	Network dump	
	Mount dump	
	Process information	Linux command 활용을 통한 정보 수집
	Network information	
	Open files	
	Loaded modules	
	Scheduled task	
	Temporary directory	

3. Linux 침해사고분석 – Overview (Cont.)

□ Overview (Cont.)

범주	세부 범주	개요
증거분석	Process information	수집된 정보를 통한 Analysis Methodology
	Network information	
	Open files	
	Loaded modules	
	침해사고조사 / 시스템 분석 시 확인 파일	침해사고조사 / 시스템 분석 시 확인해야 할 파일
	ROOTKIT 점검	Rootkit Detection Methodology
	LOG 분석	Linux Log Analysis Methodology
	의심파일 분석	Suspicious File Analysis Methodology
	Memory Dump 분석	수집된 Memory Dump 분석

3. Linux 침해사고분석 – Collecting Artifacts

□ 수집 순서

▶ 휘발성 증거 수집 후 비휘발성 증거 수집

- Memory → Storage Dump
- 대부분의 휘발성 정보는 Memory Dump에서 획득 가능
- 실제 분석은 획득한 Data(Memory/Storage Dump 등)뿐만 아니라 Live 시스템 점검과 병행하여 진행하는 것이 좋음

3. Linux 침해사고분석 – Collecting Artifacts (Cont.)

□ 휘발성 증거 수집

▶ 시스템 정보

- 시스템 시간
- 임시 데이터
- 실행중인 파일 관련 정보
  - 부모/자식 프로세스
  - 연관 정보
- 현재 로그인한 계정 정보
- 콘솔 명령어
- 메모리
- 캐시
- 로드된 모듈

▶ 네트워크 정보

- ARP 테이블
- 통신 관련 정보
- IP, DNS 설정 정보
- 통신중인 프로그램
- 네트워크 환경
- 라우팅 테이블
- 원격 사용자 정보
- 사용중인 외부 자원
- 원격 접근 파일

3. Linux 침해사고분석 – Collecting Artifacts (Cont.)

□ Memory Dump

▶ **fmem Tool을 이용한 Memory Dump 수행**

- 전체 물리메모리에 접근할 수 있는 가상의 /dev/fmem 장치 파일을 생성
- **fmem 사용법**

```
# wget http://hysteria.cz/niekt0/fmem/fmem_current.tgz
# tar -xvf fmem_current.tgz
# cd fmem*
# make
# ./run.sh
# dd if=/dev/fmem of=[output file] bs=[block size] count=[count]
# rmmod fmem
```

3. Linux 침해사고분석 – Collecting Artifacts (Cont.)

□ Memory Dump (Cont.)

▶ fmem Tool을 이용한 Memory Dump 수행 (Cont.)

- fmem 예제 (1/2)

```
[root@localhost tmp]# wget http://hysteria.cz/niekt0/fmem/fmem_current.tgz
--2018-10-10 13:14:39--  http://hysteria.cz/niekt0/fmem/fmem_current.tgz
Resolving hysteria.cz... 77.78.111.10, 2001:1528:162:20::10
Connecting to hysteria.cz|77.78.111.10|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 12566 (12K) [application/x-gzip]
Saving to: `fmem_current.tgz'

100%[======================================>] 12,566      --.-K/s   in 0s

2018-10-10 13:14:40 (545 MB/s) - `fmem_current.tgz' saved [12566/12566]

[root@localhost tmp]# tar -xvf fmem_current.tgz
fmem_1.6-0/
fmem_1.6-0/debug.h
fmem_1.6-0/README
fmem_1.6-0/ChangeLog
fmem_1.6-0/COPYING
fmem_1.6-0/AUTHORS
fmem_1.6-0/TODO
fmem_1.6-0/run.sh
fmem_1.6-0/lkm.c
fmem_1.6-0/Makefile

[root@localhost tmp]# cd fmem*
```

3. Linux 침해사고분석 – Collecting Artifacts (Cont.)

□ Memory Dump (Cont.)

▶ fmem Tool을 이용한 Memory Dump 수행 (Cont.)

- fmem 예제 (2/2)

```
[root@localhost fmem_1.6-0]# make
rm -f *.o *.ko *.mod.c Module.symvers Module.markers modules.order \.*.o.cmd \.*.ko.cmd \.*.o.d
rm -rf \.tmp_versions
make -C /lib/modules/`uname -r`/build SUBDIRS=`pwd` modules
make[1]: Entering directory `/usr/src/kernels/2.6.32-696.23.1.el6.x86_64'
  CC [M]  /tmp/fmem_1.6-0/lkm.o
  LD [M]  /tmp/fmem_1.6-0/fmem.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /tmp/fmem_1.6-0/fmem.mod.o
  LD [M]  /tmp/fmem_1.6-0/fmem.ko.unsigned
  NO SIGN [M] /tmp/fmem_1.6-0/fmem.ko
make[1]: Leaving directory `/usr/src/kernels/2.6.32-696.23.1.el6.x86_64'

[root@localhost fmem_1.6-0]# ./run.sh
Module: insmod fmem.ko al=0xffffffff81089700 : OK
Device: /dev/fmem
----Memory areas: -----
reg00: base=0x000000000 (    0MB), size= 2048MB, count=1: write-back
reg01: base=0x080000000 ( 2048MB), size= 1024MB, count=1: write-back
-----------------------
!!! Don't forget add "count=" to dd !!!

[root@localhost fmem_1.6-0]# dd if=/dev/fmem of=/tmp/fmem.mem
dd: reading `/dev/fmem': 잘못된 주소
2094952+0 records in
2094952+0 records out
1072615424 bytes (1.1 GB) copied, 13.4778 s, 79.6 MB/s

[root@localhost fmem_1.6-0]# ls -l /tmp/fmem.mem
-rw-r--r--. 1 root root 1072615424 2018-10-10 13:15 /tmp/fmem.mem

[root@localhost fmem_1.6-0]# rmmod fmem
```

3. Linux 침해사고분석 – Collecting Artifacts (Cont.)

□ Memory Dump (Cont.)

▶ **LiME Tool을 이용한 Memory Dump 수행**

- **fmem 과 동일하게 Kernel Module로 동작**
- **네트워크 전송 가능**
- **lime 사용법**

```
# git clone https://github.com/504ensicsLabs/LiME
# cd LiME/src/
# make
Local 저장          # insmod lime-*.ko path=[output path]/[output file] format=lime
Network 전송        # insmod lime-*.ko path=tcp:[port number] format=lime
# rmmod lime
```

```
Network 수신(Network 전송 시)
# nc [victim ip] [victim port] > [output file]
```

3. Linux 침해사고분석 – Collecting Artifacts (Cont.)

□ Memory Dump (Cont.)

▶ LiME Tool을 이용한 Memory Dump 수행 (Cont.)

- LiME 예제 – Local 저장

```
[root@localhost tmp]# git clone https://github.com/504ensicsLabs/LiME
Initialized empty Git repository in /tmp/LiME/.git/
remote: Enumerating objects: 242, done.
Receiving objects: 100% (242/242), 1.58 MiB | 541 KiB/s, done.
remote: Total 242 (delta 0), reused 0 (delta 0), pack-reused 242
Resolving deltas: 100% (119/119), done.

[root@localhost tmp]# cd LiME/src/

[root@localhost src]# make
make -C /lib/modules/2.6.32-696.23.1.el6.x86_64/build M="/tmp/LiME/src" modules
make[1]: Entering directory `/usr/src/kernels/2.6.32-696.23.1.el6.x86_64'
  CC [M]  /tmp/LiME/src/tcp.o
  CC [M]  /tmp/LiME/src/disk.o
  CC [M]  /tmp/LiME/src/main.o
  CC [M]  /tmp/LiME/src/hash.o
  LD [M]  /tmp/LiME/src/lime.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /tmp/LiME/src/lime.mod.o
  LD [M]  /tmp/LiME/src/lime.ko.unsigned
  NO SIGN [M] /tmp/LiME/src/lime.ko
make[1]: Leaving directory `/usr/src/kernels/2.6.32-696.23.1.el6.x86_64'
strip --strip-unneeded lime.ko
mv lime.ko lime-2.6.32-696.23.1.el6.x86_64.ko

[root@localhost src]# insmod lime-*.ko path=/tmp/mem.mem format=lime

[root@localhost src]# ls -l /tmp/mem.mem
-r--r--r--. 1 root root 1073148000 2018-10-10 10:44 /tmp/mem.mem

[root@localhost src]# rmmod lime
```

3. Linux 침해사고분석 – Collecting Artifacts (Cont.)

□ Memory Dump (Cont.)

▶ LiME Tool을 이용한 Memory Dump 수행 (Cont.)

- LiME 예제 – Network 전송 (1/2)

[피해 PC/서버]

```
[root@localhost tmp]# git clone https://github.com/504ensicsLabs/LiME
Initialized empty Git repository in /tmp/LiME/.git/
remote: Enumerating objects: 242, done.
Receiving objects: 100% (242/242), 1.58 MiB | 541 KiB/s, done.
remote: Total 242 (delta 0), reused 0 (delta 0), pack-reused 242
Resolving deltas: 100% (119/119), done.

[root@localhost tmp]# cd LiME/src/

[root@localhost src]# make
make -C /lib/modules/2.6.32-696.23.1.el6.x86_64/build M="/tmp/LiME/src" modules
make[1]: Entering directory `/usr/src/kernels/2.6.32-696.23.1.el6.x86_64'
  CC [M]  /tmp/LiME/src/tcp.o
  CC [M]  /tmp/LiME/src/disk.o
  CC [M]  /tmp/LiME/src/main.o
  CC [M]  /tmp/LiME/src/hash.o
  LD [M]  /tmp/LiME/src/lime.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /tmp/LiME/src/lime.mod.o
  LD [M]  /tmp/LiME/src/lime.ko.unsigned
  NO SIGN [M] /tmp/LiME/src/lime.ko
make[1]: Leaving directory `/usr/src/kernels/2.6.32-696.23.1.el6.x86_64'
strip --strip-unneeded lime.ko
mv lime.ko lime-2.6.32-696.23.1.el6.x86_64.ko

[root@localhost src]# insmod lime-*.ko path=tcp:55555 format=lime
```

- 조사용 PC에서 파일 수신 (다음 장 참조) -

```
[root@localhost src]# rmmod lime
```

3. Linux 침해사고분석 – Collecting Artifacts (Cont.)

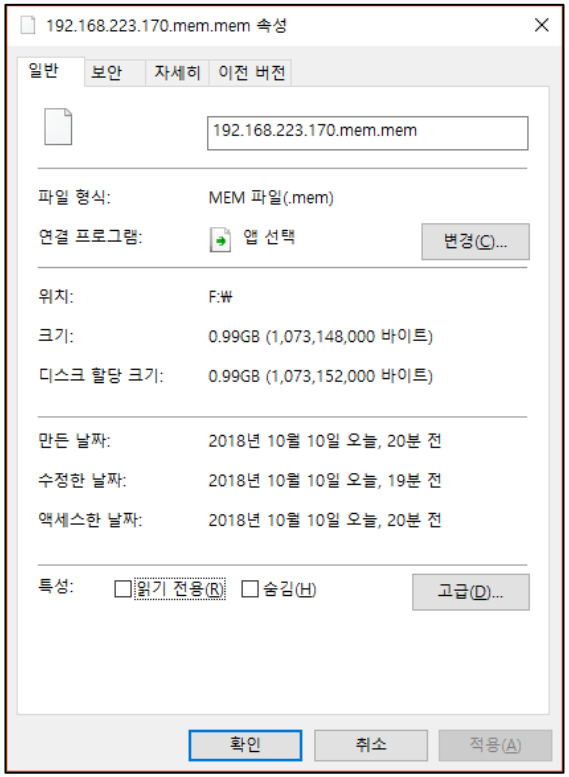
□ Memory Dump (Cont.)

▶ LiME Tool을 이용한 Memory Dump 수행 (Cont.)

- LiME 예제 – Network 전송 (2/2)

[조사용 PC]

```
F:\>nc 192.168.223.170 55555 > 192.168.223.170.mem.mem
```



3. Linux 침해사고분석 – Collecting Artifacts (Cont.)

□ Storage Dump

▶ 시스템 Shutdown 할 수 없는 경우 Live 상태에서 DD Tool을 사용하여 Dump 수행

▶ Mount 상태 확인

- fdisk -l 명령어를 통해 현재 Mount 된 Disk&Partition 정보 확인

```
Disk /dev/sda: 10.7 GB, 10737418240 bytes
255 heads, 63 sectors/track, 1305 cylinders, total 20971520 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00095ab7

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *        2048    18874367     9436160   83  Linux
/dev/sda2        18876414    20969471     1046529    5  Extended
/dev/sda5        18876416    20969471     1046528   82  Linux swap / Solaris
```

- Dump 시 Disk를 대상으로 진행

▶ Local(USB 등) 저장

- dd if=[Disk] of=[save path]/[save file name]

3. Linux 침해사고분석 – Collecting Artifacts (Cont.)

□ Storage Dump(Cont.)

▶ **Netcat tool을 사용해서 Network로 Dump 저장**

- **Dump 송/수신 시스템 모두 Netcat tool 설치 필요**

[Dump 수신(분석) 시스템]

- 임의의 Port를 열고, 저장될 파일명 지정

```
# nc -lp [Port] > [File Name]
```

예제 :

```
F:₩nc -lp 55555 > 192.168.223.170.Disk.dump
```

[Dump 송신(피해) 시스템]

- Dump 하고자 하는 Disk 선택 후 전송

```
# dd if=[Disk] | nc [Target IP] [Port]
```

예제 :

```
# dd if=/dev/sda | nc 192.168.223.1 55555
```

3. Linux 침해사고분석 – Collecting Artifacts (Cont.)

□ Process Information

▶ 명령어의 수행 결과를 Redirection 하여 Text로 저장

- ex) # command [option] > [output file]

▶ Process List

ps axf

명령어	옵션	설명
ps	-a	다른 사용자들의 실행 프로세스 출력
	-x	다른 터미널에서 실행한 프로세스 출력
	-f	Full Format 자세히 출력

▶ Process Directory

ls -alR /proc

명령어	옵션	설명
ls	-a	all 모든 정보 출력
	-l	자세히 출력
	-R	하위 디렉토리 포함 출력

3. Linux 침해사고분석 – Collecting Artifacts (Cont.)

□ Network Information

▶ 명령어의 수행 결과를 Redirection 하여 Text로 저장

- ex) # command [option] > [output file]

▶ Network Connection

netstat -antup

명령어	옵션	설명
netstat	-a	모든 소켓 보기
	-n	Hostname 없이 IP 주소 형식으로 출력
	-t	TCP 연결만 출력
	-u	UDP 연결만 출력
	-p	Program name 및 PID(Process ID) 출력

▶ ARP Cache Table

arp -e -v

명령어	옵션	설명
arp	-e	ARP 캐시테이블에 저장되어 있는 모든 호스트정보를 출력
	-v	자세한 모드로 결과를 출력 할 때 사용

▶ Routing Cache

route -C

명령어	옵션	설명
route	-C	FIB(Forwarding Information Base) 대신 Cache 출력

▶ NIC Promiscuous

ifconfig

3. Linux 침해사고분석 – Collecting Artifacts (Cont.)

□ Open Files & Loaded Modules Information

▶ 명령어의 수행 결과를 Redirection 하여 Text로 저장

- ex) # command [option] > [output file]

▶ List of Open Files

lsof -i -n -P

▶ Loaded Modules

lsmod

명령어	옵션	설명
lsof	-I	설정한 네트워크(Internet) 소켓에 대한 정보를 출력하며 프로토콜, 서비스, 호스트 및 IP에 대해 출력
	-n	호스트이름 대신에 IP address를 출력
	-P	/etc/services에 등록되어 있는 이름 대신 포트 번호로 출력

3. Linux 침해사고분석 – Collecting Artifacts (Cont.)

□ Scheduled Task

▶ 특정 프로세스나 네트워크 커넥션 정보를 관리자가 쉽게 눈치 채지 못하게 하기 위해 스케쥴을 걸어서 특정 명령을 실행 함

▶ 공격자가 시스템 장악 이후 주기적인 데몬 생성, 명령어 실행을 위해 빈번히 사용 됨

▶ 악의적인 목적에 의해 생성된 스케쥴 확인

▶ at은 한번만 실행, crontab은 주기적인 실행

3. Linux 침해사고분석 – Collecting Artifacts (Cont.)

□ Scheduled Task (Cont.)

▶ Crontab

- 현재 login 한 사용자의 Schedule 표시
- -u 옵션으로 다른 계정의 스케쥴을 볼 수 있음
- 예시

```
[root@localhost ~]# crontab -l
30 * * * * /opt/metasploit3/bin/msfupdate > /var/log/msfupdate.log 2>&1
```

```
[root@localhost ~]# crontab -u demantos -l
0 0 * * * /tmp/../rtips/rootkit_/1x/back
0 6 * * * kill -9 `ps -ef | grep demantos | awk '$NF ~ /crond/ {print $2}'`
```

- /etc 디렉토리에 시스템에서 사용되는 스케줄 작업들도 존재

```
[root@localhost ~]# ls -ld /etc/cron*
drwxr-xr-x. 2 root root 4096 2018-04-02 19:53 /etc/cron.d
drwxr-xr-x. 2 root root 4096 2018-04-02 19:54 /etc/cron.daily
-rw-r--r--. 1 root root    0 2011-07-19 13:55 /etc/cron.deny
drwxr-xr-x. 2 root root 4096 2018-04-02 19:51 /etc/cron.hourly
drwxr-xr-x. 2 root root 4096 2018-04-02 19:53 /etc/cron.monthly
drwxr-xr-x. 2 root root 4096 2011-09-27 10:33 /etc/cron.weekly
-rw-r--r--. 1 root root  457 2011-09-27 10:33 /etc/crontab
```

3. Linux 침해사고분석 – Collecting Artifacts (Cont.)

□ Scheduled Task (Cont.)

▶ Crontab (Cont.)

- 구조

항목	설명
Field 1	minute 0 ~ 59
Field 2	hour 0 ~ 23
Field 3	day 1 ~ 31
Field 4	month 1 ~ 12
Field 5	weekday 0 ~ 7 (0, 7은 일요일을 의미 1 : 월요일 2 : 화요일 ....)
Field 6	command, permmit

3. Linux 침해사고분석 – Collecting Artifacts (Cont.)

□ Temporary Directory

▶ 악의적인 목적에 의해 업로드 되거나 사용되었던 File 흔적 확인 및 채증

▶ 누구나 파일 생성이 가능한 Sticky bit 권한의 Temporary Directories

- /tmp
- /var/tmp
- /dev/shm

▶ 웹을 통해 공격할 경우 웹 권한(보통 nobody, apache, www 등)을 갖기 때문에
다른 디렉토리에 파일을 생성할 수 없지만 Sticky bit가 설정된 디렉토리에는 파일 생성 가능

▶ "..."이나 ".. "와 같은 이름을 가진 디렉토리 생성시 ls –l 명령만으로는 구분이 힘드므로
-al 옵션을 사용해서 파일 목록의 처음부분을 주의 깊게 살펴 볼 것

3. Linux 침해사고분석 – Artifact Analysis

□ 점검 항목

항목	설명
로그파일 조사	사용하지 않는 IP대역이나 비정상적인 행위를 하는 로그를 점검
	lastlog 최근 로그인 기록에서 불법적으로 접근한 사용자를 탐지
	acct, pacct 사용자의 명령로그를 점검
	sulog 불법적인 su 접근 사용자를 탐지
	wtmp 사용자의 로그인 시간 및 로그아웃 시간을 점검
	btmp 계정접속 실패 기록
	message 모든 데몬의 시스템 로그를 점검
	Xferlog 에서 ftp 접속 기록을 점검
	웹서버(access-log, errorlog) 등의 주요서비스 로그파일을 점검
	log 파일의 변조 유무 탐지

3. Linux 침해사고분석 – Artifact Analysis (Cont.)

□ 점검 항목 (Cont.)

항목	설명
파일시스템 무결성 점검	MD5등의 checksum값을 이용하여 시스템 주요 파일의 변조 여부를 조사
	redhat linux: #rpm V –a
	sun: http://sunsolve.sun.com/pub-cgi/show.pl?target=content/content7 에서 finger print 제공
루트킷 점검	chkrootkit 과 같은 루트킷 탐지 프로그램을 사용하여 점검
	시스템 주요 바이너리 변조 확인
	시스템 라이브러리 변조 확인
	커널 변조 확인
	패킷 스니퍼 점검
새로운 setuid/setgid 프로그램	주기적으로 setuid/setgid 프로그램을 주기적으로 확인하여 새로 추가된 파일이 있는지 점검
	setuid 점검 # find / -user root -perm -4000 print
	setgid 점검 # find / -group wheel -perm -2000 –print

3. Linux 침해사고분석 – Artifact Analysis (Cont.)

□ 점검 항목 (Cont.)

항목	설명
주요 설정 파일	inetd.conf, rc.sysinit, crontab 등의 시스템 주요 설정 파일의 설정 점검
	inetd.conf 에서 불법적으로 추가된 설정 점검
	rc.sysinit 파일에 백도어관련 실행파일 설정이 있는지 점검
	crontab 에서 관리목적 이외의 명령 스케줄이 포함되어 있는지 점검
	/etc/rc?.d 폴더 내 시스템 시작 시 실행 파일 점검
	추가되거나 변경된 서비스 조사
계정 점검	/etc/passwd ,shadow 파일을 점검
	사용자 홈디렉토리의 .history, .rhosts, .forward를 점검
	불법적으로 추가된 계정 점검
	권한이 상승된 계정 점검 (ex: uid가 0로 설정된 계정)
	.rhosts 파일에 불법적인 접근 서버 및 사용자 점검
	.forward 파일에 시스템 명령어 사용 점검
	.history 사용자 명령 기록에서 불법적인 접근 점검

3. Linux 침해사고분석 – Artifact Analysis (Cont.)

□ 점검 항목 (Cont.)

항목	설명
열린 포트 탐지	비정상적인 포트나 일반적인 접속이 아닌 접속자의 IP및 서비스 포트를 점검
	netstat 를 통한 접속자의 IP및 접속 포트 등의 네트웍 정보 점검
	nmap을 통한 열린 포트 점검 ex) nmap P0 p1-65535 xxx.xxx.xxx.xxx
	lsof를 통한 열린 포트 탐지하여 정상상태와 비교
프로세스 탐지	ps를 이용하여 현재 실행중인 프로세스들의 이상 유무를 판별
	불법적인 프로세스 동작 여부 점검 (ex: 2개 이상의 inetd)
	이상한 이름의 프로세스 조사 (ex: exploit.sh)
sniffing 탐지	ifconfig나 chkrootkit을 이용 네트웍 스니핑 탐지
보안패치 점검	새로운 보안 패치가 존재하는지 점검

3. Linux 침해사고분석 – Artifact Analysis (Cont.)

□ MAC TIME 분석

▶ 침해사고분석 시 가장 중요한 부분은 시간으로 MAC Time에 대한 이해가 필요

▶ atime (access time)

- 파일 접근 시간
- write, read, access 할 때 마다 업데이트 되며, CLI, system process 에 의해 수행 될 수 있음

▶ mtime (modify time)

- 파일 내용 수정 시간
- 파일의 권한 변경으로는 update 되지 않음. 파일 내용 변경 추적에 용의 함

▶ ctime (change time)

- 파일 속성이 변경 된 시간
- 소유권, 액세스 권한 또는 파일 수정 시 업데이트
- 파일의 상태 정보의 최종 수정 시간, 아이 노드의 데이터가 변경이 될 때마다 update

3. Linux 침해사고분석 – Artifact Analysis (Cont.)

□ MAC TIME 분석 (Cont.)

▶ 파일 시간 속성

```
[root@localhost mac]# date
2018. 10. 10. (수) 17:29:52 KST                          // 현재 날짜 확인
[root@localhost mac]# touch test1                        // 파일 생성
[root@localhost mac]# stat test1                         // 생성 파일 정보 확인
  File: `test1'
  Size: 0                Blocks: 0          IO Block: 4096   일반 빈 파일
Device: fd00h/64768d     Inode: 930378      Links: 1
Access: (0644/-rw-r--r--)  Uid: (    0/    root)   Gid: (    0/    root)
Access: 2018-10-10 17:29:56.752265874 +0900              // atime, ctime, mtime 동일
Modify: 2018-10-10 17:29:56.752265874 +0900
Change: 2018-10-10 17:29:56.752265874 +0900
[root@localhost mac]# cat > test1                        // 파일 내용 추가
test1
^Z
[3]+  Stopped                 cat > test1
[root@localhost mac]# stat test1                         // 파일 내용 수정으로 mtime, ctime 변경
  File: `test1'
  Size: 6                Blocks: 8          IO Block: 4096   일반 파일
Device: fd00h/64768d     Inode: 930378      Links: 1
Access: (0644/-rw-r--r--)  Uid: (    0/    root)   Gid: (    0/    root)
Access: 2018-10-10 17:29:56.752265874 +0900
Modify: 2018-10-10 17:30:08.250249102 +0900
Change: 2018-10-10 17:30:08.250249102 +0900
```

3. Linux 침해사고분석 – Artifact Analysis (Cont.)

□ MAC TIME 분석 (Cont.)

▶ 파일 시간 속성 (Cont.)

```
[root@localhost mac]# vi test1                          // 파일 내용 수정(사이즈 동일)
[root@localhost mac]# stat test1                        // File Open 및 내용 수정으로 atime, mtime, ctime 변경
  File: `test1'
  Size: 6               Blocks: 8          IO Block: 4096   일반 파일
Device: fd00h/64768d    Inode: 930378      Links: 1
Access: (0644/-rw-r--r--)  Uid: (    0/    root)   Gid: (    0/    root)
Access: 2018-10-10 17:30:25.689826681 +0900
Modify: 2018-10-10 17:30:33.588919997 +0900
Change: 2018-10-10 17:30:33.589920263 +0900
[root@localhost mac]# chmod 600 test1                   // 파일 권한 변경
[root@localhost mac]# stat test1                        // 파일 속성 변경으로 ctime만 변경
  File: `test1'
  Size: 6               Blocks: 8          IO Block: 4096   일반 파일
Device: fd00h/64768d    Inode: 930378      Links: 1
Access: (0600/-rw-------)  Uid: (    0/    root)   Gid: (    0/    root)
Access: 2018-10-10 17:30:25.689826681 +0900
Modify: 2018-10-10 17:30:33.588919997 +0900
Change: 2018-10-10 17:31:00.329092563 +0900
```

3. Linux 침해사고분석 – Artifact Analysis (Cont.)

□ MAC TIME 분석 (Cont.)

▶ 파일 시간 속성 (Cont.)

```
[root@localhost mac]# touch -t 201810080000 test1        // touch 로 강제 시간 변경
[root@localhost mac]# stat test1                         // 강제 변경한 시간이 ctime에 기록
  File: `test1'
  Size: 6                Blocks: 8          IO Block: 4096   일반 파일
Device: fd00h/64768d     Inode: 930378      Links: 1
Access: (0600/-rw-------)  Uid: (    0/    root)   Gid: (    0/    root)
Access: 2018-10-08 00:00:00.000000000 +0900
Modify: 2018-10-08 00:00:00.000000000 +0900
Change: 2018-10-10 17:31:52.775492240 +0900
[root@localhost mac]# mv test1 test2                     // 파일명 변경
[root@localhost mac]# stat test2                         // 파일 속성만 변경되어 ctime만 변경
  File: `test2'
  Size: 6                Blocks: 8          IO Block: 4096   일반 파일
Device: fd00h/64768d     Inode: 930378      Links: 1
Access: (0600/-rw-------)  Uid: (    0/    root)   Gid: (    0/    root)
Access: 2018-10-08 00:00:00.000000000 +0900
Modify: 2018-10-08 00:00:00.000000000 +0900
Change: 2018-10-10 17:32:08.374398221 +0900
```

3. Linux 침해사고분석 – Artifact Analysis (Cont.)

□ MAC TIME 분석 (Cont.)

▶ File Systems and Time Stamping

File System	Access Time	Change Time	Modify Time	Creation Time
ext2	○	○	○	X
ext3	○	○	○	X
ext4	○	○	○	○
ntfs	○	○	○	○

▶ Update Example

Command	Access Time	Modify Time	Change Time
mv	-	-	update
cp	update	-	-
touch	update	update	update
redirection	-	update	update
change owner	-	-	update

3. Linux 침해사고분석 – Artifact Analysis (Cont.)

□ MAC TIME 분석 (Cont.)

▶ File Creation Time

- ext4의 경우 Creation Time이 있으나 stat 명령어로 확인 불가

```
[root@localhost tmp]# stat /tmp/fmem.mem
  File: `/tmp/fmem.mem'
  Size: 1072615424  Blocks: 2094960    IO Block: 4096   일반 파일
Device: fd00h/64768d    Inode: 935151      Links: 1
Access: (0644/-rw-r--r--)  Uid: (    0/    root)   Gid: (    0/    root)
Access: 2018-10-10 13:15:33.174073671 +0900
Modify: 2018-10-10 13:15:46.649202704 +0900
Change: 2018-10-10 13:15:46.649202704 +0900
```

- System debugger 사용으로 확인 가능

```
[root@localhost tmp]# debugfs -R 'stat /tmp/fmem.mem' /dev/mapper/VolGroup-lv_root
debugfs 1.41.12 (17-May-2010)
Inode: 935151   Type: regular    Mode:  0644   Flags: 0x80000
Generation: 1865285449    Version: 0x00000000:00000001
User:     0   Group:     0   Size: 1072615424
File ACL: 0    Directory ACL: 0
Links: 1   Blockcount: 2094960
Fragment:  Address: 0    Number: 0    Size: 0
 ctime: 0x5bbd7cf2:9ac83040 -- Wed Oct 10 13:15:46 2018
 atime: 0x5bbd7ce5:29809d1c -- Wed Oct 10 13:15:33 2018
 mtime: 0x5bbd7cf2:9ac83040 -- Wed Oct 10 13:15:46 2018
crtime: 0x5bbd7ce5:29809d1c -- Wed Oct 10 13:15:33 2018
Size of extra inode fields: 28
```

- 명령어

√ # debugsfs –R ‘stat [file]’ /dev/[partition]

3. Linux 침해사고분석 – Artifact Analysis (Cont.)

□ 시간 기반의 검색

▶ ls

- list 명령어인 ls의 경우 옵션으로 시간 검색이 가능함
- c 옵션 change time, u 옵션 access time, 옵션이 없을 경우 modify time 표시

```
ls –tl : mtime(생성, 내용변경 시간) 조회
ls –ul : atime(최종 접근시간) 조회
ls –cl : ctime(i-node 변경시간) 조회
```

▶ find

- ndays 옵션으로 number를 기준으로 수정된 파일을 검색 할 수 있음
  - √ # find [PATH] -ndays [+/-]number -print

```
# find / -atime –ndays –ls : ndays 이전 시점부터 현재까지 access된 모든 파일을 검색
# find / -mtime –ndays –ls : ndays 이전 시점부터 현재까지 변경된 모든 파일을 검색
# find / -ctime –ndays –ls : ndays 이전 시점부터 현재까지 inode 변경된 모든 파일을 검색
# find . –mtime 0 –print : 오늘 생성, 혹은 내용 수정된 파일 찾기
# find . –mtime 1 –print : 어제 생성, 혹은 내용 수정된 파일 찾기
# find . –mtime -3 –print : (오늘을 빼고) 3일내에 생성, 혹은 내용 수정된 파일 찾기
# find . –mtime +3 –print : 생성, 혹은 내용 수정된지 3일이 넘는 파일 찾기
# find . –mtime +5 –mtime -12 –print : 생성, 혹은 내용 수정된지 5일은 넘고 12일은 안되는 파일 찾기
# find . –type f –atime -2 –print : 2일내에 접근한 적이 있는 파일 찾기
# find /dev –type f –ls (/dev 디렉토리는 device에 관련된 많은 파일들이 존재)
# find / -name ".*" –print –xdev | cat –v 또는 find / -name ".. " –print –xdev 등의 명령으로 침입자가 숨겨둔 파일 검색
# find / -perm –4000 –user root –print 또는 find / -perm –2000 –group root –print 등의 명령으로 침입자가 시스템 숨겨둔
백도어(backdoor) 파일을 검색
```

3. Linux 침해사고분석 – Artifact Analysis (Cont.)

□ Process Information 분석

▶ Process List

- ps 명령어는 다양한 옵션을 가지고 있으므로 다양한 결과를 도출할 수 있음
- 좌측 그림 적색칸 안에서 5436은 PID(ProcessID), 2405는 PPID(ParentPID)를 의미
- 좌측 그림에서는 부모 프로세스를 찾아 가야 하는 단점이 있지만 우측 그림은 트리 구조
- 현재 실행 중인 프로세스 중 의심스러운 프로세스가 있는지 확인

```
test      2245     1  0 Mar17 ?        00:00:00 /usr/lib/zeitgeist/zeit
test      2246     1  0 Mar17 ?        00:00:01 zeitgeist-datahub
test      2248  2207  0 Mar17 ?        00:00:00 gnome-pty-helper
test      2250  2207  0 Mar17 pts/2    00:00:00 bash
test      2252  2245  0 Mar17 ?        00:00:00 /bin/cat
test      2338  1928  0 Mar17 ?        00:00:01 telepathy-indicator
test      2339     1  0 Mar17 ?        00:00:00 /usr/lib/unity-lens-mus
test      2359     1  0 Mar17 ?        00:00:00 /usr/lib/telepathy/miss
test      2363     1  0 Mar17 ?        00:00:00 /usr/lib/gnome-online-a
test      2368     1  0 Mar17 ?        00:00:00 /usr/bin/python /usr/li
test      2386  1928  0 Mar17 ?        00:00:06 gnome-screensaver
root      2397  2250  0 Mar17 pts/2    00:00:00 su -
root      2405  2397  0 Mar17 pts/2    00:00:00 -su
test      2467  1928  0 Mar17 ?        00:00:03 update-notifier
test      2518  1928  0 Mar17 ?        00:00:00 /usr/lib/deja-dup/deja-
test      2523     1  0 Mar17 ?        00:00:00 /usr/bin/python /usr/li
root      2890     1  0 Mar17 ?        00:00:00 /usr/bin/python /usr/li
root      3535   317  0 Mar17 ?        00:00:00 /sbin/udevd --daemon
test      4454  2207  0 01:06 pts/3    00:00:00 bash
root      4509  4454  0 01:07 pts/3    00:00:00 su -
root      4517  4509  0 01:07 pts/3    00:00:00 -su
root      4935     1  0 02:06 ?        00:00:00 ./ishd -i 65535 -t 0 -p
root      4936  4935  0 02:06 ?        00:00:00 sh -c /bin/sh
root      4937  4936  0 02:06 ?        00:00:00 /bin/sh
root      5436  2405  0 03:24 pts/2    00:00:00 ./attack -l -p 666
lp        6813   540  0 07:40 ?        00:00:00 /usr/lib/cups/notifier/
root      9483     2  0 17:36 ?        00:00:00 [flush-8:0]
root      9495     2  0 17:38 ?        00:00:00 [kworker/0:0]
root      9516     2  0 17:43 ?        00:00:00 [kworker/0:2]
```

[ps –ef]

```
 837 ?        Ss     0:00 acpid
 841 ?        Ss     0:00 cron
 842 ?        Ss     0:00 atd
 846 ?        Ssl    0:00 whoopsie
 950 ?        Ssl    0:00 lightdm
 994 tty7     Ss+    2:30  \_ Xorg
1831 ?        Sl     0:00  \_ lightdm
1928 ?        Ssl    0:01      \_ gnome-session
1963 ?        Ss     0:00          \_ ssh-agent
1979 ?        Sl     0:09          \_ gnome-settings-
1994 ?        Sl     3:58          \_ compiz
2112 ?        Ss     0:00          |   \_ sh
2113 ?        Sl     0:05          |   |   \_ gtk-window-deco
2206 ?        Ss     0:00          |   \_ sh
2207 ?        Sl     0:18          |       \_ gnome-terminal
2248 ?        S      0:00          |           \_ gnome-pty-helpe
2250 pts/2    Ss     0:00          |           \_ bash
2397 pts/2    S      0:00          |           |   \_ su
2405 pts/2    S      0:00          |           |       \_ bash
5436 pts/2    S+     0:00          |           |           \_ attack
4454 pts/3    Ss     0:00          |           \_ bash
4509 pts/3    S      0:00          |               \_ su
4517 pts/3    S      0:00          |                   \_ bash
9584 pts/3    R+     0:00          |                       \_ ps
2013 ?        Sl     0:05          \_ nautilus
2014 ?        Sl     0:01          \_ bluetooth-apple
2015 ?        Sl     0:01          \_ nm-applet
2018 ?        Sl     0:01          \_ gnome-fallback-
2028 ?        Sl     0:01          \_ polkit-gnome-au
```

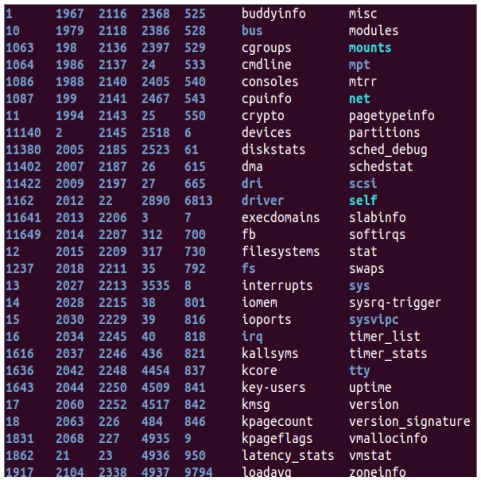
[ps axfc]

3. Linux 침해사고분석 – Artifact Analysis (Cont.)

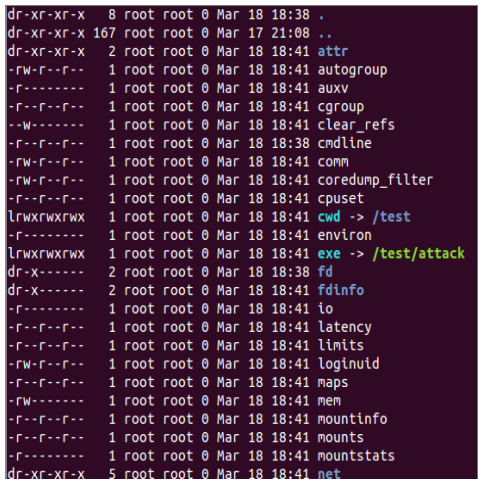
□ Process Information 분석 (Cont.)

▶ Process Directory

- /proc directory는 process, memory 등 시스템 실시간 데이터에 대한 정보가 파일 형태로 저장된 가상 파일 시스템
- Disk에 저장 되지 않고 Kernel에 의해 메모리에 적재 됨
- 좌측 그림의 숫자로 된 directory는 process number이고, 우측 그림은 directory 내부이며 실행 된 실제 경로 등의 확인 가능



[/proc]



[/proc 하위 Directory]

3. Linux 침해사고분석 – Artifact Analysis (Cont.)

□ Network Information 분석

▶ Network Connection

- netstat 명령을 이용하여 비정상 접속이 있는지를 확인
- netstat 명령 옵션 중 p 옵션은 해당 소켓의 PID와 프로그램 이름을 출력
  - √ 이 정보를 통해 /proc 디렉토리에서 좀 더 많은 정보 획득 가능
- 'netstat –antup' 명령어를 이용한 분석

```
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.1:53            0.0.0.0:*               LISTEN      1087/dnsmasq
tcp        0      0 127.0.0.1:631           0.0.0.0:*               LISTEN      540/cupsd
tcp        0      0 0.0.0.0:666             0.0.0.0:*               LISTEN      5436/attack
tcp        1      0 192.168.100.164:47090   91.189.89.144:80        CLOSE_WAIT  2187/ubuntu-geoip-p
tcp6       0      0 ::1:631                 :::*                    LISTEN      540/cupsd
udp        0      0 0.0.0.0:5353            0.0.0.0:*                           528/avahi-daemon: r
udp        0      0 0.0.0.0:56657           0.0.0.0:*                           528/avahi-daemon: r
udp        0      0 127.0.0.1:53            0.0.0.0:*                           1087/dnsmasq
udp        0      0 0.0.0.0:68              0.0.0.0:*                           700/dhclient
udp6       0      0 :::56545                :::*                                528/avahi-daemon: r
udp6       0      0 :::5353                 :::*                                528/avahi-daemon: r
```

3. Linux 침해사고분석 – Artifact Analysis (Cont.)

□ Network Information 분석 (Cont.)

▶ ARP Cache Table

- ARP Cache 테이블에는 해당 시스템과 연결되었던 내부 시스템의 IP와 MAC address가 저장되고 이를 통해 어떤 시스템과 통신 했는지 확인이 가능
- 또는 ARP Cache 테이블 결과 중 서로 다른 IP에서 동일한 MAC address가 있는지 확인
- 'arp –e –v' 명령어를 이용한 점검

Address	HWtype	HWaddress	Flags Mask	Iface
192.168.100.2	ether	00:50:56:fa:40:4a	C	eth0
192.168.100.11	ether	00:0c:29:b8:5e:15	C	eth0
192.168.100.254	ether	00:50:56:e7:5e:2c	C	eth0

3. Linux 침해사고분석 – Artifact Analysis (Cont.)

□ Network Information 분석 (Cont.)

▶ Routing Cache

- route 명령어를 이용하여 Use 항목을 통해 사용 빈도를 유추 해 볼 수 있음.
- 특정 IP에서 너무 많은 사용이 있었을 경우 스캐닝성 공격을 의심 할 수 있음
- 'route –c' 명령어를 이용한 점검

```
Kernel IP routing cache
Source          Destination     Gateway         Flags Metric Ref    Use Iface
192.168.100.1   192.168.100.255 192.168.100.255 ibl   0      0     1183 lo
192.168.100.254 ubuntu.local    ubuntu.local    il    0      0        0 lo
ubuntu.local    mistletoe.canon 192.168.100.2         0      1        2 eth0
ubuntu.local    192.168.100.254 192.168.100.254       0      0        0 eth0
192.168.100.2   ubuntu.local    ubuntu.local    il    0      0        0 lo
192.168.100.1   192.168.100.255 192.168.100.255 ibl   0      0     1183 lo
192.168.100.254 ubuntu.local    ubuntu.local    il    0      0        0 lo
ubuntu.local    mistletoe.canon 192.168.100.2         0      1        2 eth0
ubuntu.local    gremlin.canonic 192.168.100.2         0      0        0 eth0
kns.kornet.net  ubuntu.local    ubuntu.local    l     0      0        6 lo
```

3. Linux 침해사고분석 – Artifact Analysis (Cont.)

□ Network Information 분석 (Cont.)

▶ NIC Promiscuous

- NIC(Network Interface Card)가 sniffing mode로 동작하는지 확인
- NIC는 자신에게만 오는 패킷만 수신하지만 Promiscuous mode(sniffing mode) 시 다른 NIC로 가는 패킷도 수신 함
- 'ifconfig –a'명령어를 이용한 점검

```
eth0      Link encap:Ethernet  HWaddr 00:0c:29:47:ff:c4
          inet addr:192.168.[redacted]  Bcast:192.168.[redacted]  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe47:ffc4/64 Scope:Link
          UP BROADCAST RUNNING PROMISC MULTICAST  MTU:1500  Metric:1
          RX packets:4831591 errors:0 dropped:0 overruns:0 frame:0
          TX packets:9590292 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:339142071 (339.1 MB)  TX bytes:3723774968 (3.7 GB)
          Interrupt:19 Base address:0x2024
```

3. Linux 침해사고분석 – Artifact Analysis (Cont.)

□ Open Files & Loaded Modules Information 분석

▶ List of Open Files

- 현재 Open 되어 있는 File 의 목록. 특정 process 가 사용하고 있는 File 을 확인 할 수 있음
- 'lsof'명령어를 이용한 점검

```
lsof: WARNING: can't stat() fuse.gvfs-fuse-daemon file system /home/test/.gvfs
      Output information may be incomplete.
COMMAND     PID      USER   FD      TYPE     DEVICE SIZE/OFF       NODE NAME
init          1         0  cwd       DIR        8,1     4096          2 /
init          1         0  rtd       DIR        8,1     4096          2 /
init          1         0  txt       REG        8,1   194528         77 /sbin/init
init          1         0  mem       REG        8,1    47040     134730 /lib/i386-linux-gnu/libnss_files-2.15.so
init          1         0  mem       REG        8,1    42652     134734 /lib/i386-linux-gnu/libnss_nis-2.15.so
init          1         0  mem       REG        8,1    92016     134724 /lib/i386-linux-gnu/libnsl-2.15.so
init          1         0  mem       REG        8,1    30520     134726 /lib/i386-linux-gnu/libnss_compat-2.15.so
init          1         0  mem       REG        8,1  1713640     134679 /lib/i386-linux-gnu/libc-2.15.so
init          1         0  mem       REG        8,1    30684     134765 /lib/i386-linux-gnu/librt-2.15.so
init          1         0  mem       REG        8,1   124663     134759 /lib/i386-linux-gnu/libpthread-2.15.so
init          1         0  mem       REG        8,1   292132     134691 /lib/i386-linux-gnu/libdbus-1.so.3.5.8
```

[lsof]

```
attack    9148       root  cwd       DIR        8,1     4096     131354 /test
attack    9148       root  rtd       DIR        8,1     4096          2 /
attack    9148       root  txt       REG        8,1    22080     131606 /test/attack
attack    9148       root  mem       REG        8,1    47040     134730 /lib/i386-linux-gnu/libnss_files-2.15.so
attack    9148       root  mem       REG        8,1  1713640     134679 /lib/i386-linux-gnu/libc-2.15.so
attack    9148       root  mem       REG        8,1   134344     134659 /lib/i386-linux-gnu/ld-2.15.so
attack    9148       root    0u      CHR      136,2      0t0          5 /dev/pts/2
attack    9148       root    1u      CHR      136,2      0t0          5 /dev/pts/2
attack    9148       root    2u      CHR      136,2      0t0          5 /dev/pts/2
attack    9148       root    3u     IPv4      90110      0t0        TCP *:5020 (LISTEN)
```

[lsof에서 grep attack 결과]

3. Linux 침해사고분석 – Artifact Analysis (Cont.)

□ Open Files & Loaded Modules Information 분석 (Cont.)

▶ Loaded Modules

- 악성 module도 정상적인 module name으로 로드 된 경우가 많음
- 정상적인 module의 경우 대부분 /lib/modules/[kernel version] 하위 directory 에 존재 함

```
Module                  Size  Used by
vmhgfs                 53736  0
vsock                  39001  0
acpiphp                23535  0
vmwgfx                102138  2
ttm                    65344  1 vmwgfx
drm                   197692  4 vmwgfx,ttm
snd_ens1371            24819  2
gameport               15060  1 snd_ens1371
snd_ac97_codec        106082  1 snd_ens1371
ac97_bus               12642  1 snd_ac97_codec
snd_pcm                80845  2 snd_ens1371,snd_ac97_codec
snd_seq_midi           13132  0
snd_rawmidi            25424  2 snd_ens1371,snd_seq_midi
snd_seq_midi_event     14475  1 snd_seq_midi
snd_seq                51567  2 snd_seq_midi,snd_seq_midi_event
snd_timer              28931  2 snd_pcm,snd_seq
snd_seq_device         14172  3 snd_seq_midi,snd_rawmidi,snd_seq
psmouse                72846  0
joydev                 17393  0
snd                    62064  11 snd_ens1371,snd_ac97_codec,snd_pcm
awmidi,snd_seq,snd_timer,snd_seq_device
vmw_balloon            12700  0
serio_raw              13027  0
bnep                   17830  2
rfcomm                 38139  0
bluetooth             158438  10 bnep,rfcomm
parport_pc             32114  1
```

[lsmod]

```
root@ubuntu:~# modinfo ttm
filename:       /lib/modules/3.2.0-23-generic-pae/kernel/drivers/gpu/drm/
ttm/ttm.ko
license:        GPL and additional rights
description:    TTM memory manager subsystem (for DRM device)
author:         Thomas Hellstrom, Jerome Glisse
srcversion:     576484675D37456C29D23D7
depends:        drm
intree:         Y
vermagic:       3.2.0-23-generic-pae SMP mod_unload modversions 686
```

[modinfo [module name]]

3. Linux 침해사고분석 – Artifact Analysis (Cont.)

□ 침해사고조사 / 시스템 분석 시 확인 파일

▶ Account File

- /etc/passwd, /etc/shadow, /etc/group
- 의도하지 않은, 또는 알 수 없는 user 의 생성
- root 계정 이외에 uid 또는 gid가 0 인 계정
- /bin/bash 또는 /bin/sh 등 쉘 사용이 가능한 계정
- /etc/shadow 파일에 패스워드가 설정된 계정 중 시스템 계정 존재 확인

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mailing List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bi
```

[/etc/passwd]

```
root:$6$PVRV7L2N$KGKHc49Jfytel1Ke3WVTS1LFtkfWPBuAdzGjzgA6p9ZCBEQt9
820:0:99999:7:::
daemon:*:15453:0:99999:7:::
bin:*:15453:0:99999:7:::
sys:*:15453:0:99999:7:::
sync:*:15453:0:99999:7:::
games:*:15453:0:99999:7:::
man:*:15453:0:99999:7:::
lp:*:15453:0:99999:7:::
mail:*:15453:0:99999:7:::
news:*:15453:0:99999:7:::
uucp:*:15453:0:99999:7:::
proxy:*:15453:0:99999:7:::
www-data:*:15453:0:99999:7:::
backup:*:15453:0:99999:7:::
list:*:15453:0:99999:7:::
```

[/etc/shadow]

3. Linux 침해사고분석 – Artifact Analysis (Cont.)

□ 침해사고조사 / 시스템 분석 시 확인 파일 (Cont.)

▶ Startup Script

- 부팅 시 기본으로 설정된 run level directory 확인
- 보통의 경우 run level 3(text mode)의 경우가 많음
- 관리자가 특별히 setting 하지 않는 경우 파일의 생성 시간은 대부분 동일
- 각 file 앞 K는 Kill을 S는 Start 를 의미
- 대문자만 정상 동작, 소문자는 정상 동작 하지 않음
- 보통 관리자들이 추가로 실행할 서비스들을 rc.local 파일에 등록

```
drwxr-xr-x 2 root root 4096  9월  3  2013 /etc/rc0.d
drwxr-xr-x 2 root root 4096  9월  3  2013 /etc/rc1.d
drwxr-xr-x 2 root root 4096  9월  3  2013 /etc/rc2.d
drwxr-xr-x 2 root root 4096  9월  3  2013 /etc/rc3.d
drwxr-xr-x 2 root root 4096  9월  3  2013 /etc/rc4.d
drwxr-xr-x 2 root root 4096  9월  3  2013 /etc/rc5.d
drwxr-xr-x 2 root root 4096  9월  3  2013 /etc/rc6.d
-rwxr-xr-x 1 root root  306  4월 23  2012 /etc/rc.local
drwxr-xr-x 2 root root 4096 11월  1  2012 /etc/rcS.d
```

[/etc/rc*]

```
-rw-r--r--   1 root root   677  4월 14  2012 README
lrwxrwxrwx   1 root root    20  9월 11  2012 S20kerneloops -> ../init.d/kerneloops
lrwxrwxrwx   1 root root    27  9월 11  2012 S20speech-dispatcher -> ../init.d/speech-dispatcher
lrwxrwxrwx   1 root root    23  9월  3  2013 S20uml-utilities -> ../init.d/uml-utilities
lrwxrwxrwx   1 root root    20  9월 11  2012 S50pulseaudio -> ../init.d/pulseaudio
lrwxrwxrwx   1 root root    15  9월 11  2012 S50rsync -> ../init.d/rsync
lrwxrwxrwx   1 root root    15  9월 11  2012 S50saned -> ../init.d/saned
lrwxrwxrwx   1 root root    19  9월 11  2012 S70dns-clean -> ../init.d/dns-clean
lrwxrwxrwx   1 root root    18  9월 11  2012 S70pppd-dns -> ../init.d/pppd-dns
lrwxrwxrwx   1 root root    14  9월 11  2012 S75sudo -> ../init.d/sudo
lrwxrwxrwx   1 root root    20  1월 30  2013 S91aolserver4 -> ../init.d/aolserver4
lrwxrwxrwx   1 root root    17 12월  6  2012 S91apache2 -> ../init.d/apache2
lrwxrwxrwx   1 root root    22  9월 11  2012 S99acpi-support -> ../init.d/acpi-support
lrwxrwxrwx   1 root root    21  9월 11  2012 S99grub-common -> ../init.d/grub-common
lrwxrwxrwx   1 root root    18  9월 11  2012 S99ondemand -> ../init.d/ondemand
lrwxrwxrwx   1 root root    18  9월 11  2012 S99rc.local -> ../init.d/rc.local
```

[/etc/rc5.d]

3. Linux 침해사고분석 – Artifact Analysis (Cont.)

□ 침해사고조사 / 시스템 분석 시 확인 파일 (Cont.)

▶ Super Daemon

- inetd 또는 xinetd 라 함.
- inetd 경우 /etc/inet.conf, xinet 경우 /etc/xinetd.d 에 각 서비스에 대한 설정 파일 존재
- 각 설정 파일에 disable 항목이 no 일 경우 사용이 되고 있는 부분이니 확인이 필요함
- 백도어 설치에 유용하게 사용

```
# default: on
# description: The telnet server serves telnet sessions; it uses ₩
# unencrypted username/password pairs for authentication.
service telnet
{
        flags = REUSE
        socket_type = stream
        wait = no
        user = root
        server = /bin/sh -i
        disable = no
}
```

```
[root@Cent ~]# grep telnet /etc/services
telnet 8080/tcp
telnet 8080/udp
rtelnet 107/tcp # Remote Telnet
rtelnet 107/udp
telnets 992/tcp
telnets 992/udp
skytelnet 1618/tcp # skytelnet
...[skip]...
```

3. Linux 침해사고분석 – Artifact Analysis (Cont.)

□ 침해사고조사 / 시스템 분석 시 확인 파일 (Cont.)

▶ .rhosts , /etc/hosts.equiv
- 각 계정 홈디렉토리와 /etc 디렉토리에 각각 존재
- global 설정 파일은 /etc/hosts.equiv 로 존재
- 해당 파일은 신뢰된 원격 호스트 이름 또는 IP가 명시 되어 있음
- 명시된 호스트는 인증 절차 없이 내부 접근 가능

```
hostname [username]
```

- hostname이나 username에 “+” 가 있을 경우 any를 뜻하고 “+ +”는 누구든 인증 절차 없이 접근할 수 있다는 것을 의미

▶ /etc/hosts
- 특정 IP에 대한 hostname의 별칭을 매칭 해 놓은 파일
- 비정상적인 매칭 구문이 있는지 파악

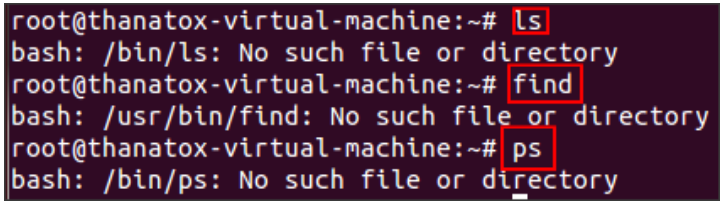
▶ /etc/resolv.conf
- DNS 서버 저장
- /etc/hosts 파일 우선 이후 /etc/resolv.conf 파일 확인 (기본 설정)

3. Linux 침해사고분석 – Artifact Analysis (Cont.)

□ ROOTKIT 점검

▶ ROOTKIT
- 탐지를 피하기 위해 사용 되는 프로그램의 종류
- 시스템 실행 파일의 변경 또는 시스템 라이브러리를 교체, 커널 모듈에 인스톨 등

▶ ROOTKIT 현상
- syslogd 데몬 종료
- 시스템 기본 유틸리티 교체
- ssh, telnet, finger 데몬 추가
- 백그라운드로 sniffer 작동



[ROOTKIT 감염 시스템]

3. Linux 침해사고분석 – Artifact Analysis (Cont.)

□ ROOTKIT 점검 (Cont.)

▶ chkrootkit

Install

```
# make sense
# chkrootkit : -q 옵션 사용시 의심가는 파일들만 출력
# chkproc –v : ps명령으로 PID가 보이지 않는 프로세스들을 검출
```

사용방법

- chkrootkit : shell script that checks system binaries for rootkit modification.
- ifpromisc.c : checks if the interface is in promiscuous mode.
- chklastlog.c : checks for lastlog deletions.
- chkwtmp.c : checks for wtmp deletions.
- check_wtmpx.c : checks for wtmpx deletions. (Solaris only)
- chkproc.c : checks for signs of LKM trojans.
- chkdirs.c : checks for signs of LKM trojans.
- strings.c : quick and dirty strings replacement.
- chkutmp.c : checks for utmp deletions.

3. Linux 침해사고분석 – Artifact Analysis (Cont.)

□ ROOTKIT 점검 (Cont.)

▶ rkhunter

Install

./installer --install 또는 # ./installer.sh

update : rkhunter의 경우 최신 버전으로 업데이트가 자체적으로 가능함. (network 사용)

rkhunter --update

rkhunter -c

rkhunter.conf 파일에서 일부 옵션 들을 활성 또는 비활성 할 수 있으므로 install 전에 필요 옵션이 있는지 확인 필요

사용방법

- -c : --checkall 옵션
- -sk : --skip-keypress 결과 리포트중에 확인 키입력을 받지 않는다.
- --update : 업데이트
- --createlogfile : /var/log/rkhunter.log에 로그를 남긴다.
- --cronjob : 클론탭 모드로 동작한다(컬러레이아웃을 제거한다.)
- --report-warnings-olny : 문제가 되는 사항만 출력한다.

3. Linux 침해사고분석 – Artifact Analysis (Cont.)

□ ROOTKIT 점검 (Cont.)

▶ unhide

- 루트킷이나 LKM에 의해 숨겨진 프로세스나 TCP/UDP 포트를 찾아준다.
- 총 6가지 기법을 이용하여 숨겨진 프로세스를 찾는다.

√ Compare /proc vs /bin/ps output

√ Compare info gathered from /bin/ps with info gathered by walking thru the procfs.

√ Compare info gathered from /bin/ps with info gathered from syscalls (syscall scanning).

√ Full PIDs space occupation (PIDs bruteforcing)

√ Reverse search, verify that all thread seen by ps are also seen by the kernel (/bin/ps output vs /proc, procfs walking and syscall)

√ Quick compare /proc, procfs walking and syscall vs /bin/ps output.

3. Linux 침해사고분석 – Artifact Analysis (Cont.)

□ ROOTKIT 점검 (Cont.)

▶ STRACE

- ROOTKIT 검사 도구를 통해 확인된 PROCESS의 PID를 이용하여 TRACE

strace 사용법

```
# strace -fr -p 1234   --> 기본 호출 방법
# strace -ftt -o trace.log -p 1234
```

사용방법

-f 부모와 자식 프로세스 모두 추적 (항상 넣어주는 것이 좋다)
-tt 작업 수행한 시각 표시
-ttt 작업 수행한 시각 표시(epoch time)
-r 시스템 호출 사이에 흐른 시간 (현재tt - 이전tt)
-T 시스템 호출에 걸린 시간만 표시
-p PID
-c 시스템 호출 요약 정보 표시 (프로세스 정지할 때까지 멈출 수 없다)
-o 결과를 파일로 출력

3. Linux 침해사고분석 – Artifact Analysis (Cont.)

□ Log 분석

▶ Log 종류

로그	형식	설명	명령어
utmp, utmpx	바이너리	현재 로그인한 사용자의 상태 정보	w , who , user , finger
wtmp, wtmpx	바이너리	사용자의 로그인 , 로그아웃 정보	last
dmesg, boot	텍스트	시스템 부팅 시 관련 정보	dmesg
secure , auth.log	텍스트	사용자 인증 관련 정보	
btmp, failedlogin	텍스트	로그인 실패 정보	lastb
maillog	텍스트	메일 데몬 관련 정보	
lastlog	바이너리	사용자의 최근 로그인 정보	lastlog
messages	텍스트	시스템 서비스 및 성능 관련 정보	
xferlog	텍스트	FTP 서비스 송수신 관련 정보	
pacct	바이너리	각 사용자별 프로세스 , 명령어 정보	lastcomm
access_log	텍스트	웹 서비스 접근 정보	
error_log	텍스트	웹 서비스 에러 정보	

3. Linux 침해사고분석 – Artifact Analysis (Cont.)

□ Log 분석 (Cont.)

▶ 삭제된 Log

- rootkit과 같은 자동화된 툴에 의해 로그가 조작되거나 파일은 존재하는데 로그 내용이 삭제되는 경우가 있는데 이런 경우 chkrootkit의 chkwtmp, chklastlog, chkutmp 등의 툴을 이용하여 어떤 시점의 로그가 삭제되었는지 확인이 가능

```
[root@Cent chkrootkit-0.49]# ./chkwtmp
1 deletion(s) between Mon May  7 00:40:59 2012 and Mon May  7 10:45:32 2012
1 deletion(s) between Mon May  7 10:47:36 2012 and Mon May  7 10:58:35 2012
1 deletion(s) between Mon Aug 13 16:54:20 2012 and Wed May 15 16:06:24 2013
[root@Cent chkrootkit-0.49]#
[root@Cent chkrootkit-0.49]# ./chklastlog
user root deleted or never logged from lastlog!
[root@Cent chkrootkit-0.49]#
[root@Cent chkrootkit-0.49]# ./chkutmp
=> possibly 1 deletion(s) detected in /var/run/utmp !
 The tty of the following user process(es) were not found
 in /var/run/utmp !
! RUID          PID TTY    CMD
! root         3819 tty1   /sbin/mingetty tty1
! root         3820 tty2   /sbin/mingetty tty2
! root         3821 tty3   /sbin/mingetty tty3
! root         3822 tty4   /sbin/mingetty tty4
! root         3865 tty7   /usr/bin/Xorg :0 -br -audit 0 -auth /var/gdm/:0.Xauth -nolisten tcp vt7
! root         6354 pts/1  -bash
```

3. Linux 침해사고분석 – Artifact Analysis (Cont.)

□ Log 분석 (Cont.)

▶ 삭제된 Log (Cont.)

- 특정 툴을 사용하지 않고 파일의 시간(Access, Modify, Change Time)을 통해서 어느 정도 추정이 가능
- 로그파일의 시간이 모두 동일한 경우 의심.
- 보통 분할된 로그 파일의 mtime은 해당 로그 파일의 마지막에 로그가 기록된 시간과 거의 일치

```
[root@localhost log]# stat secure-20181010
  File: `secure-20181010'
  Size: 925286      Blocks: 1816       IO Block: 4096   일반 파일
Device: fd00h/64768d    Inode: 791479      Links: 1
Access: (0600/-rw-------)  Uid: (    0/    root)   Gid: (    0/    root)
Access: 2018-04-02 19:57:46.671921787 +0900
Modify: 2018-10-10 15:15:30.074087955 +0900
Change: 2018-10-10 15:26:00.879782580 +0900
You have new mail in /var/spool/mail/root

[root@localhost log]# tail secure-20181010
Oct 10 13:13:48 localhost pam: gdm-password[3032]: pam_unix(gdm-password:session): session open
Oct 10 13:13:48 localhost polkitd(authority=local): Unregistered Authentication Agent for sessi
Oct 10 13:13:51 localhost polkitd(authority=local): Registered Authentication Agent for session
Oct 10 13:14:33 localhost su: pam_unix(su-l:session): session opened for user root by infosec(u
Oct 10 14:36:29 localhost su: pam_unix(su-l:session): session opened for user root by infosec(u
Oct 10 14:46:10 localhost gnome-screensaver-dialog: pam_unix(gnome-screensaver:auth): authentic
Oct 10 14:46:10 localhost gnome-screensaver-dialog: gkr-pam: unlocked 'login' keyring
Oct 10 15:15:22 localhost unix_chkpwd[8586]: password check failed for user (root)
Oct 10 15:15:22 localhost su: pam_unix(su-l:auth): authentication failure; logname=infosec uid=
Oct 10 15:15:30 localhost su: pam_unix(su-l:session): session opened for user root by infosec(u
```

3. Linux 침해사고분석 – Artifact Analysis (Cont.)

□ Log 분석 (Cont.)

▶ HITORY LOG

- 공격자가 쉘 접근을 했을 경우 히스토리 로그를 통해 어떤 행동을 했는지 확인 가능
- 단, Reverse Connection을 맺어 쉘을 사용할 경우 히스토리 로그에서 실행했던 명령을 확인할 수 없다.
- 히스토리 파일을 /dev/null로 링크를 걸어 놨을 경우에도 확인할 수 없다.
- User 별로 각 home directory 내에 text 형태로 존재

```
[root@localhost log]# find / -regextype posix-egrep -iregex "(₩/₩w+)+₩/₩.((ba|c)?sh_)?history"

/root/.bash_history
/home/infosechack/.bash_history
/home/infosec/.bash_history
```

3. Linux 침해사고분석 – Artifact Analysis (Cont.)

□ Log 분석 (Cont.)

▶ HITORY LOG (Cont.)

- basic setting 에서는 시간 정보가 기록 되지 않음
- 시간 정보 삽입 및 확인 방법

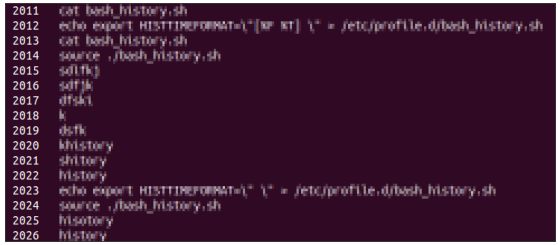
√ shell script 작성

```
# echo export HISTTIMEFORMAT=₩"[%F %T] ₩" > /etc/profile.d/bash_history.sh
```

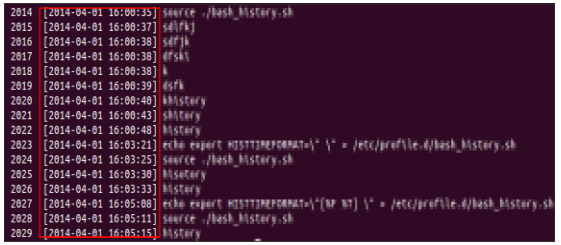
√ /etc/profile 에 추가

```
HISTTIMEFORMAT='%Y-%m-%d %H:%M:%S'
export HISTTIMEFORMAT
```

- 재 로그인 후 적용이 되나 source 명령어로 강제 적용 가능



[Format 적용 전]



[Format 적용 후]

3. Linux 침해사고분석 – Artifact Analysis (Cont.)

□ Log 분석 (Cont.)

▶ Message LOG

- /var/log/messages 에 위치하며, text 형태로 저장
- 설정의 오류 메세지 등도 저장
- 'root' 권한으로의 불법적인 로그인 시도가 있었는지를 살펴본다.
- 'su' 명령을 이용한 'root' 또는 특정 권한의 사용자로의 의심스러운 전환시도 확인
- 유효한 사용자로부터의 반복적인 실패한 로그인 시도 확인

```
[root@Cent ~]# more  messages
Dec  4 16:23:54 msd1 su: 'su root' failed for aster on /dev/pts/1
Dec  4 17:14:30 msd1 rsh[10103]: connection from bad port
Dec  4 17:24:39 msd1 su: 'su root' failed for aster on /dev/pts/2
Dec  4 17:24:48 msd1 last message repeated 1 time
Dec  5 11:39:29 msd1 su: 'su root' failed for aster on /dev/pts/1
Dec  5 20:57:52 msd1 syslogd: going down on signal 15
```

3. Linux 침해사고분석 – Artifact Analysis (Cont.)

□ Log 분석 (Cont.)

▶ Authentication LOG

- Telnet, FTP, SSH 등 인증 과정을 거치는 모든 행위에 대한 로그를 저장
- 보통 secure라는 파일명, 시스템에 따라 auth.log라는 파일명으로 저장
- 일반적으로 로그인 성공, 실패 로그가 저장되고 syslog 데몬의 시작/중지 로그도 저장
- 로그인 실패 기록이 많을 경우 Brute-Force 공격을 했을 것으로 추정 가능,
  로그인 실패 이후 동일한 IP에서 로그인 성공 기록까지 확인 필요

3. Linux 침해사고분석 – Artifact Analysis (Cont.)

□ Log 분석 (Cont.)

▶ wtmp/btmp Log

- wtmp는 last(로그인 기록), btmp는 lastb(로그인 실패 기록)
- 바이너리 형태로 저장되며 텍스트 에디터로는 확인이 불가, last 명령을 통해서 확인 가능

```
reboot   system boot  3.2.0-23-generic Sun Mar 23 18:54 - 18:54  (00:00)
test     pts/3        :0.0             Tue Mar 18 01:06 - crash (5+17:47)
test     pts/2        :0.0             Mon Mar 17 21:09 - crash (5+21:45)
reboot   system boot  3.2.0-23-generic Mon Mar 17 21:08 - 18:54 (5+21:46)
test     pts/0        :0.0             Mon Mar 17 19:29 - 19:30  (00:00)
test     pts/0        :0.0             Mon Mar 17 19:28 - 19:29  (00:00)
reboot   system boot  3.2.0-23-generic Tue Mar 18 03:47 - 19:30  (-8:-16)

wtmp begins Tue Mar 18 03:47:02 2014
```

- 구조

구분	설명
Filed 1	user
Filed 2	connection terminal
Filed 3	connection IP
Filed 4	login time
Filed 5	logout time
Filed 6	duration

3. Linux 침해사고분석 – Artifact Analysis (Cont.)

□ Log 분석 (Cont.)

▶ lastlog Log

- 각 계정(/etc/passwd에 등록된 계정)의 마지막 로그인 기록
- 바이너리 형태로 저장되며 텍스트 에디터로는 확인이 불가, lastlog 명령을 통해서 확인 가능

```
root@localhost tmp]# lastlog
사용자이름         포트      어디서              최근정보
root                                          **한번도 로그인한 적이 없습니다**
bin                                           **한번도 로그인한 적이 없습니다**
daemon                                        **한번도 로그인한 적이 없습니다**
adm                                           **한번도 로그인한 적이 없습니다**
lp                                            **한번도 로그인한 적이 없습니다**
sync                                          **한번도 로그인한 적이 없습니다**
shutdown                                      **한번도 로그인한 적이 없습니다**
halt                                          **한번도 로그인한 적이 없습니다**
mail                                          **한번도 로그인한 적이 없습니다**
uucp                                          **한번도 로그인한 적이 없습니다**
operator                                      **한번도 로그인한 적이 없습니다**
games                                         **한번도 로그인한 적이 없습니다**
gopher                                        **한번도 로그인한 적이 없습니다**
ftp                                           **한번도 로그인한 적이 없습니다**
nobody                                        **한번도 로그인한 적이 없습니다**
dbus                                          **한번도 로그인한 적이 없습니다**
usbmuxd                                       **한번도 로그인한 적이 없습니다**
avahi-autoipd                                 **한번도 로그인한 적이 없습니다**
vcsa                                          **한번도 로그인한 적이 없습니다**
rpc                                           **한번도 로그인한 적이 없습니다**
rtkit                                         **한번도 로그인한 적이 없습니다**
abrt                                          **한번도 로그인한 적이 없습니다**
saslauth                                      **한번도 로그인한 적이 없습니다**
postfix                                       **한번도 로그인한 적이 없습니다**
qpidd                                         **한번도 로그인한 적이 없습니다**
haldaemon                                     **한번도 로그인한 적이 없습니다**
ntp                                           **한번도 로그인한 적이 없습니다**
apache                                        **한번도 로그인한 적이 없습니다**
avahi                                         **한번도 로그인한 적이 없습니다**
rpcuser                                       **한번도 로그인한 적이 없습니다**
nfsnobody                                     **한번도 로그인한 적이 없습니다**
pulse                                         **한번도 로그인한 적이 없습니다**
gdm                                           **한번도 로그인한 적이 없습니다**
sshd                                          **한번도 로그인한 적이 없습니다**
tcpdump                                       **한번도 로그인한 적이 없습니다**
infosec                                       **한번도 로그인한 적이 없습니다**
tomcat                                        **한번도 로그인한 적이 없습니다**
mysql                                         **한번도 로그인한 적이 없습니다**
infosechack      pts/0    192.168.223.148   목  4월  5 15:41:06 +0900 201
```

3. Linux 침해사고분석 – Artifact Analysis (Cont.)

□ Log 분석 (Cont.)

▶ su 관련 Log

- secure log 내 su 명령 확인

```
May 16 10:56:36 Cent su: pam_unix(su-l:session): session opened for user demantos by root(uid=0)
May 16 10:56:37 Cent su: pam_unix(su-l:session): session closed for user demantos
May 16 10:56:51 Cent su: pam_unix(su-l:session): session opened for user demantos by root(uid=0)
May 16 10:57:04 Cent su: pam_unix(su-l:auth): authentication failure; logname=root uid=500 euid=0 tty=pts/3
ruser=demantos rhost= user=root
May 16 10:57:08 Cent su: pam_unix(su-l:session): session closed for user demantos
May 16 10:57:23 Cent su: pam_unix(su-l:session): session opened for user demantos by root(uid=0)
May 16 10:57:26 Cent su: pam_unix(su-l:session): session opened for user root by root(uid=500)
May 16 10:57:35 Cent su: pam_unix(su-l:session): session closed for user root
May 16 10:57:37 Cent su: pam_unix(su-l:session): session closed for user demantos
May 16 10:58:13 Cent su: pam_unix(su-l:session): session opened for user demantos by root(uid=0)
May 16 10:58:16 Cent su: pam_unix(su-l:session): session opened for user root by root(uid=500)
May 16 10:58:17 Cent su: pam_unix(su-l:session): session closed for user root
May 16 10:58:18 Cent su: pam_unix(su-l:session): session closed for user demantos
```

- w 명령으로 utmp Log 확인

```
[root@Cent ~]# w
 11:27:59 up 24 min, 3 users, load average: 0.00, 0.01, 0.06
USER TTY FROM LOGIN@ IDLE JCPU PCPU WHAT
root :0 - 11:05 ?xdm? 11.35s 0.27s /usr/bin/gnome-session
root pts/1 :0.0 11:06 19:24 0.03s 0.03s bash
root pts/2 192.168.126.1 11:27 1.00s 0.02s 0.02s w
```

3. Linux 침해사고분석 – Artifact Analysis (Cont.)

□ Log 분석 (Cont.)

▶ top Log

- 오랜 시간 수행되고 있는 process
- 시스템에서 사용하지 않는 명칭의 process
- CPU 점유율이 높은 process

```
[root@Cent ~]# top
Load averages : 1.68, 1.50, 11:53:26
182 processes: 2 running, 32 sleeping, 147 idle, 1 zombie
CPU states:  7.2% user,  9.0% nice, 83.7% system,  0.0% idle
Memory: Real: 261M/367M act/tot  Virtual: 532M use/tot  Free: 44M
  PID USERNAME PRI NICE  SIZE   RES STATE   TIME    CPU COMMAND
31119  aster    57    2 1768K  204K run    68:46 89.30% rsh
19119  aster    44    0 2768K  491K run     0:00  0.70% top
  484  root     44    0 1784K  155K sleep  64:06  0.40% inetd
 5060  aster    44    0 2176K  294K sleep   0:28  0.30% ck2
  584  root     44    0 6272K 2662K sleep   4:21  0.00% insightd
   26  root     44    0 1704K   90K sleep   2:38  0.00% update
  818  root     42   -2 8992K 3178K sleep   1:00  0.00% Xdec
 2493  acle     44    0   18M 2621K sleep   0:37  0.00% oracle
  920  root     44    0   10M 1630K sleep   0:37  0.00% dtsession
  148  root     44    0 1744K  172K sleep   0:34  0.00% syslogd
30837  aster    44    0 2176K  294K sleep   0:29  0.00% ck
 2499  acle     44    0   18M 3235K sleep   0:20  0.00% oracle
  966  root     44    0 6640K  835K sleep   0:16  0.00% ttsession
 2492  acle     44    0   18M  712K sleep   0:16  0.00% oracle
 2494  acle     44    0   18M  729K sleep   0:07  0.00% oracle
```

3. Linux 침해사고분석 – Artifact Analysis (Cont.)

□ 의심 파일 분석

▶ dev

- /dev 디렉토리 하위에는 대부분 장치 파일들만 존재
- /dev/MAKEDEV나 /dev/.udev 디렉토리는 정상 파일
- 루트킷에서 /dev 디렉토리에 파일을 생성하는 경우 발생

```
[root@Cent ~]# find /dev -type f
...[skip]...
/dev/.udev/db/class@usb_device@usbdev1.1
/dev/.udev/db/class@usb_device@usbdev2.1
/dev/.udev/db/class@usb_device@usbdev2.2
/dev/.udev/db/class@usb_device@usbdev2.3
/dev/.udev/db/class@input@input2@mouse1
/dev/.udev/db/class@input@input3@mouse2
/dev/.udev/db/block@ram0
/dev/.udev/db/block@ram1
/dev/.udev/db/class@misc@device-mapper
/dev/.udev/db/class@input@mice
/dev/.udev/db/class@input@input1@mouse0
/dev/.udev/uevent_seqnum
/dev/shm/swapcache_cbec
```

3. Linux 침해사고분석 – Artifact Analysis (Cont.)

□ 의심 파일 분석 (Cont.)

▶ Hidden files and directories

- 파일명 또는 디렉토리명 앞에 점(.)을 붙일 경우 숨김 속성 설정

```
[root@Cent ~]# find / -name ".*"
/etc/skel/.bash_logout
/etc/skel/.mozilla
/etc/skel/.bashrc /etc/skel/.bash_profile
/etc/lvm/cache/.cache /etc/.pwd.lock
/sys/module/autofs4/sections/.strtab
/sys/module/autofs4/sections/.symtab
/sys/module/autofs4/sections/.module_sig
/sys/module/autofs4/sections/.bss
...(skip)...
```

3. Linux 침해사고분석 – Artifact Analysis (Cont.)

□ 의심 파일 분석 (Cont.)

▶ Hidden files and directories (Cont.)

- 파일에 숨김 속성을 설정하기 보다는 ".. "과 같은 형태로 디렉토리에 숨김 속성을 설정

```
[root@Cent tmp]# ls -al
total 192
drwxr-xr-x 2 root root 4096 Jan 2 17:12
drwxrwxrwt 18 root root 4096 Jan 2 17:12 .
drwxr-xr-x 2 root root 4096 Jan 2 17:12 .
drwxr-xr-x 24 root root 4096 May 7 2012 ..
drwxr-xr-x 2 root root 4096 Jan 2 17:12 ..
drwxr-xr-x 3 root root 4096 Sep 19 16:26 ...
-rw------- 1 demantos root 9 Jan 2 16:17 crontab.XXXXcuAOtO
drwxrwxrwt 2 root root 4096 May 7 2012 .font-unix
drwx------ 3 root root 4096 May 7 2012 gconfd-root
srw-rw-rw- 1 root root 0 May 7 2012 .gdm_socket
-rw------- 1 root root 115 Aug 13 16:53 .gdmYOM8DW
drwxrwxrwt 2 root root 4096 May 7 2012 .ICE-unix
drwx------ 2 root root 4096 May 7 2012 keyring-TAF4Nv
-rwx------ 1 root root 1384 May 6 2012 ks-script-_E8NU8
-rw-r--r-- 1 root root 70 May 6 2012 ks-script-_E8NU8.log
srwxr-xr-x 1 root root 0 May 7 2012 mapping-root
drwx------ 2 root root 4096 Aug 13 18:48 orbit-root
```

3. Linux 침해사고분석 – Artifact Analysis (Cont.)

□ 의심 파일 분석 (Cont.)

▶ Hidden files and directories (Cont.)

- 숨김 파일/디렉토리 검색

```
[root@Cent tmp]# find / -name " *" -ls
3139702 8 drwxr-xr-x 2 root root 4096 Jan 2 17:12 /tmp/₩ ₩ ₩
[root@Cent tmp]# find / -name "..*" -ls
2068949 8 -rw-r--r-- 1 root root 40 Jul 22 2011 /usr/share/man/man1/..1.gz
3139706 8 drwxr-xr-x 2 root root 4096 Jan 2 17:12 /tmp/..₩
3106916 8 drwxr-xr-x 3 root root 4096 Sep 19 16:26 /tmp/...
```

```
[root@Cent tmp]# find / -name ".*" -type d -ls
...[snip]...
2683054 8 drwxr-xr-x 3 root root 4096 Aug 13 19:36 /usr/share/.xx
3237766 8 drwxr-xr-x 4 demantos demantos 4096 May 6 2012 /home/demantos/.mozilla
3139706 8 drwxr-xr-x 2 root root 4096 Jan 2 17:12 /tmp/..₩
3139705 8 drwxr-xr-x 2 root root 4096 Jan 2 17:12 /tmp/.₩
3107009 8 drwxrwxrwt 2 root root 4096 May 7 2012 /tmp/.X11-unix
3106916 8 drwxr-xr-x 3 root root 4096 Sep 19 16:26 /tmp/...
3139692 8 drwxr-xr-x 2 root root 4096 Sep 19 16:27 /tmp/.../rtips/infect_/grep.src/lib/.deps
3139660 8 drwxr-xr-x 2 root root 4096 Sep 19 16:27 /tmp/.../rtips/rootkit_/1x/.tmp_versions
3107005 8 drwxrwxrwt 2 root root 4096 May 7 2012 /tmp/.font-unix
3106882 8 drwxrwxrwt 2 root root 4096 May 7 2012 /tmp/.ICE-unix
```

3. Linux 침해사고분석 – Artifact Analysis (Cont.)

□ Memory Dump 분석

▶ 메모리 분석의 장점

- 암호화/패킹된 실행 파일의 경우 메모리상에서는 복호화되어 존재
- 실제 파일이 실행되어야 메모리에 로드되기 때문에 실행 여부 확인 가능
- 은닉된 프로세스나 네트워 정보 확인 가능
- 분석의 반복성 보장
- 기타 대부분의 휘발성 정보를 메모리 덤프를 통해 획득할 수 있음

3. Linux 침해사고분석 – Artifact Analysis (Cont.)

□ Memory Dump 분석 (Cont.)

▶ Volatility

- 메모리 분석용 open-source tool로 windows와 linux를 모두 지원
- 기본 문법은 아래와 같음 (linux version 기준)

  √ python ./vol.py –f [dump image 파일명] --profile=[OS profile 종류] plugin명
- Profile/Plugin 확인

  √ python ./vol.py --info

  √ Linux System에서 사용가능한 Plugin은 Plugin명 앞에 Linux_ 가 있음
- 주요 플러그인

항목	설명
linux_netstat	Lists open sockets
linux_ifconfig	Gathers active interfaces
linux_arp	Print the ARP table
linux_lsmod	Gather loaded kernel modules
linux_lsof	Lists open files
linux_malfind	Looks for suspicious process mappings
linux_mount	Gather mounted fs/devices

3. Linux 침해사고분석 – Artifact Analysis (Cont.)

□ Memory Dump 분석 (Cont.)

▶ Volatility (Cont.)

- Linux의 경우 Profile이 존재하지 않는 경우가 있음, 이 경우 다운로드/직접 생성 필요

  √ Volatility Homepage에서 다운로드(https://github.com/volatilityfoundation/profiles)

  √ Volatility 설치 후 아래의 명령어로 생성

```
# cd [volatility directory]/tools/linux/
# make ( module.dwarf 이 생성)
# zip [profile name] ./module.dwarf /boot/[make 이후 system map file]
- make를 진행하면 module.dwarf 파일과 system map file이 생성되고, 이 파일을 zip으로 묶는 작업
- 완료 되면 해당 파일은 [volatility directory]/volatility/plugins/overlays/linux 에 위치 시킴.
```

```
Profiles
--------
LinuxUbuntu12x86  - A Profile for Linux Ubuntu12 x86
VistaSP0x64       - A Profile for Windows Vista SP0 x64
VistaSP0x86       - A Profile for Windows Vista SP0 x86
VistaSP1x64       - A Profile for Windows Vista SP1 x64
VistaSP1x86       - A Profile for Windows Vista SP1 x86
VistaSP2x64       - A Profile for Windows Vista SP2 x64
VistaSP2x86       - A Profile for Windows Vista SP2 x86
Win2003SP0x86     - A Profile for Windows 2003 SP0 x86
Win2003SP1x64     - A Profile for Windows 2003 SP1 x64
Win2003SP1x86     - A Profile for Windows 2003 SP1 x86
```

3. Linux 침해사고분석 – Linux File Recovery

□ 파일 시스템 복구

▶ Ext2

- Ext2은 특정 파일을 삭제 시 실제 파일은 데이터 블록에 그대로 위치하며 I-node 테이블 정보로만 삭제되어 복구가 가능함.

  √ Debugfs 툴을 사용하여 복구 가능

```
# debugfs /dev/[디스크파티션]
# cd /[디렉토리]
# lsdel      (삭제된 파일 및 디렉토리 출력)
# stat <i-node Number>      (i-node 정보 출력)
# dump <i-node Number> /[디렉토리]/[파일]      (삭제된 파일 복구 명령)
```

```
root@topcert-virtual-machine:/home/topcert# debugfs /dev/sda1
debugfs 1.44.1 (24-Mar-2018)
debugfs:
debugfs:
debugfs:
```

```
debugfs:
debugfs:  lsdel

 Inode  Owner  Mode    Size      Blocks   Time deleted
0 deleted inodes found.
(END)
```

3. Linux 침해사고분석 – Linux File Recovery

□ 파일 시스템 복구

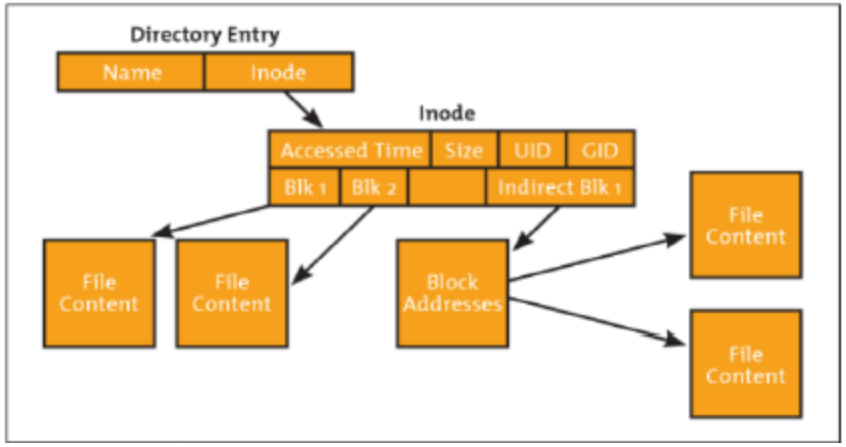
▶ Ext3, Ext4

- Ext3, Ext4은 Journaling 으로 인해 특정 파일을 삭제 시 메타데이터가 I-node에서 삭제

  √ 더 이상 파일의 내용이 어디에 있는지 알 수 없게 됨

- 저널링 기법을 이용하여 파일을 복구

  √ I-node의 엔트리 블록이 저널에 기록된다는 것을 이용하여 복구



3. Linux 침해사고분석 – Linux File Recovery

□ 파일 시스템 복구

▶ Ext3

- Ext3 파일 시스템은 Ext3grep 툴을 이용하여 복구 가능

  √ 설치 : Apt-get install ex3grep

  √ 매뉴얼 : Man ext3grep

```
# ext3grep /dev/[디바이스] --dump-names
# ext3grep /dev/[디바이스] --restore-all
# ext3grep /dev/[디바이스] --restore-file
( 결과 : RESTORE_FILES 저장 )
```

3. Linux 침해사고분석 – Linux File Recovery

□ 파일 시스템 복구

▶ Ext4

- Ext4 파일 시스템은 Extundelete 툴을 이용하여 복구 가능

√ 설치방법

```
# apt-get install g++
# apt-get install e2fslibs-dev
# wget https://sourceforge.net/projects/extundelete/files/extundelete/0.2.4/extundelete-0.2.4.tar.bz2
# tar –xvf extundelete-0.2.4.tar.bz2
# cd extundelete-0.2.4.tar.bz2
# ./configure       (/usr/local/bin 으로 설치 됨)
# make && make install
```

```
# exutndelete /dev[디바이스] --restore-directory /tmp
# exutndelete /dev[디바이스] --restore-all
# exutndelete /dev[디바이스] --restore-files /tmp/del.jpg
# exutndelete /dev[디바이스] --i-node [i-node 넘버], [i-node 넘버], ....
( 결과 : RESTORE_FILES 저장 )
```

3. Linux 침해사고분석 – Linux File Recovery

□ 이미지 덤프를 이용한 복구

▶ 소프트웨어 방식

- Disk를 사전에 준비하여 소프트웨어를 이용하여 디스크 덤프
- 3.5 inchi HardDrive + Box

- 2.5 inchi Portable Hard Drive

3. Linux 침해사고분석 – Linux File Recovery

□ 이미지 덤프를 이용한 복구

▶ 하드웨어 방식

- Logicube Falcon

- Logicube Dossier

3. Linux 침해사고분석 – Linux File Recovery

□ 이미지 덤프를 이용한 복구

▶ 이미지에서 파일 데이터 복구

√ 대표적인 데이터 복구 도구

Encase

Forensic Explorer

R-Studio

√ 파일 시그니처 검색하여 복구

복구하려는 파일의 정확한 속성 이해

Linux 혹은 Unix의 경우 최근 CCTV, IOT 등에 많이 사용

사용되는 용도에 따라 파일 시그니처가 다른 경우가 많음

3. Linux 침해사고분석 – Trend

□ 암호화 (Encryption)

▶ 공격 진행, 유출 구간 암호화

√ 공격에 사용하는 악성코드, 웹쉘 암호화/난독화

√ Directory 이동, 파일 복사 등 명령 구간 자체를 암호화 하며 침해

명령어 SET을 암호화하여 저장된 파일을 업로드

암호 해제 용 파일이나 명령어로 대상 실행

√ 공개된 암호 알고리즘 사용

RC4, DES, AES 등

사용된 알고리즘을 확인하면 Key 형태 유추 가능

Key 확보 시 해제 가능

√ 자체 암호화 루틴 사용

Key의 형식이나 사용법 알기 어려움

사용되는 코드, 악성코드 확보 후 분석 시 해제 가능

√ Steganography

공격 혹은 정보 유출 시 그림 파일 등에 암호화된 정보 저장

3. Linux 침해사고분석 – Trend

□ Linux 통신 은닉

▶ 자체 통신 프로토콜 구현

√ TCP/IP 프로토콜이 자동으로 처리하는 헤더 제어

일반적인 포맷을 사용하지 않고 자체 제작하여 패킷 주고 받음

IP와 포트 상관 없이 모든 패킷 스니핑 가능 (ex. Wireshark)

RAW Socket 이라고 함

√ 백도어로 악용

포트를 열고 있어도 netstat 에 탐지되지 않음

명령을 주고 받아도 흔적이 남지 않음

3. Linux 침해사고분석 – Trend

□ 정상 프로그램/명령 악용

▶ 악성코드가 아닌 정상 프로그램/명령을 이용하여 해킹

√ 시스템에 존재하는 기능 적극 활용

서버 관리도구, 도메인 관리도구 등 관리 툴 악용

로깅 설정이 잘 되지 않는 Powershell 적극 활용

공격자 행위와 사용자 행위가 섞여 판단을 어렵게 함

√ Open Source 프로그램 변조

관리자가 Windows -> Linux 접근 시 별도 프로그램 사용 (ex. SSH)

Open Source 접속 프로그램 변조, 명령 실행 결과 출력을 다른 곳으로 우회

Linux 에 흔적이 남지 않아 공격자의 행위 파악 어려움

√ Web Source 변조

피해 시스템의 웹 소스 변조하여 웹쉘 기능 생성

정상 파일들과 섞이고 변조 때문에 분석 시간 증가

3. Linux 침해사고분석 – Trend

□ Fileless Attack

▶ 공격 흔적이 메모리에만 남는 수준의 공격

√ 정상 어플리케이션 활용

취약점 존재하는 어플리케이션 버전을 받거나 다운그레이드 후 공격 (ex. IE)

Batch, Powershell 등 script 로 한번에 원하는 악성 행위 수행

√ Fileless Malware

필요 시 메모리에 악성코드 적재, 로더 자체는 악성코드가 아닌 경우

Windows 의 경우 Registry에 악성코드 은닉하는 사례 증가

Linux 의 경우 CLI 특성상 분석가의 시선을 흐리는 프로세스 명으로 변조 후 자가 삭제

√ 메모리 포렌식의 어려움

사고 직후 Forensic 절차에 따르는 경우가 드뭄

많은 피해 시스템들이 재부팅 되는 경우가 많음

3. Linux 침해사고분석 – Trend

□ 공격자들의 목표 다변화

▶ 고도화된 APT 증가 추세

√ 금전 획득

Ransomware, Miner 설치는 공격자의 주된 목표

비교적 목표가 뚜렷하고 탐지가 쉬워 분석 및 조치가 용이함

√ 금전 탈취 & 정보유출

APT로 진행되는 경우가 많음

조사 기간이 광범위하고 대상이 많음

해당 기업 Target 으로 제작된 전용 악성코드 다수 존재

Anti-Forensic 기법 다수 활용

외부 기간과 연계 필요

◈ 실습

실습 시간

◈ 마치며

감사합니다.