



# BPFDoor 악성코드 점검 가이드

작성자: 한국인터넷진흥원 위협분석단 종합분석팀

배포일: 2025년 5월 12일



## 주의사항

해당 가이드는 Linux 시스템에서 'BPFDoor 악성코드' 탐지를 지원하기 위한 명령어와 스크립트를 포함하고 있습니다. 다만, 시스템 별 환경 차이(커널 버전, 보안 설정, 서비스 구성 등)로 인해 해당 명령어나 스크립트 실행 시, 시스템에 예기치 않은 동작 또는 오류가 발생할 수 있습니다.

따라서 실행 전 ▲사내 보안 정책 위반 여부, ▲시스템 영향 가능성 등을 반드시 확인해 주시기 바라며, 실행으로 인해 발생하는 모든 결과 및 책임은 사용자 본인에게 있음을 안내 드립니다.

## 개요

- 최근 BPFDoor 악성코드 감염 사례가 발견됨에 따라, 기업 내 보안 담당자가 스스로 감염 여부를 점검하고 조기에 대응할 수 있도록 본 문서를 작성하였습니다.
- 본 문서는 악성코드의 특성을 분석하여, 단순한 침해 지표(IOC) 검색만으로는 확인하기 어려운 감염 흔적까지 점검할 수 있도록 가이드를 제공합니다.

## BPFDoor 악성코드

- 리눅스 환경에서 포트를 열지 않고 외부 연결을 대기하는 백도어 악성코드
  - 위협 행위자의 매직패킷(Magic Packet) 수신 시 셸 연결
    - 리버스셸(Reverse Shell), 다이렉트 모드(바인드셸, Bind Shell) 연결
    - 명령에 따라 원격지(Remote)와 셸 연결
- BPF 기술을 악용하여 네트워크 트래픽 필터 설정
  - 필터 대상 : TCP, UDP, ICMP 프로토콜
  - 감염 시스템에 합법적으로 열린 포트(정상 서비스) 대상 매직패킷 전송



### BPF(Berkeley Packet Filter)

- 운영체제 커널 수준에서 동작하는 네트워크 패킷 필터링 기술
- 네트워크 인터페이스를 통과하는 패킷을 복사하여 필터링
- 네트워크 성능 향상과 보안 기능 강화를 위한 목적으로 사용

## BPFDoor 감염 여부 점검 방법

- 점검 명령어는 관리자(root) 계정 또는 권한(sudo)으로 실행해야 합니다.
- 점검 과정에서 침해 흔적 발견 시 한국인터넷진흥원에 신고하시기 바랍니다.

### 1. 악성코드 뮉텍스/락(Mutex/Lock) 파일 점검 : ls 명령어

- 특정 경로에 생성되는 뮉텍스/락 파일을 점검하여 악성코드 실행 여부를 점검합니다.
  - 점검 대상 : /var/run/\*.pid 또는 /var/run/\*.lock (0 byte)
    - 파일 권한(644, -rw-r--r--) 확인
  - ⚠ 파일 존재 시 다른 점검 방법으로 추가 검증 필요

```
# 명령어 1-1 : *.pid, *.lock 파일 확인
$ sudo ls -l /var/run/*.pid | awk '$5 == 0 {print $9}'
$ sudo ls -l /var/run/*.lock | awk '$5 == 0 {print $9}'
$ sudo stat -c "%a %s %n" /var/run/*.pid /var/run/*.lock 2>/dev/null | awk '$1=="644" && $2==0 { print $3 }'
```

```
# 명령어 1-2 : 파일 권한(644,-rw-r--r--), 파일 생성일 확인
$ sudo ls -al /var/run/<filename>.pid
-rw-r--r--. 1 root root 0 Apr 28 03:17 /var/run/system.pid
```

### 2. 악성코드 자동 실행 파일 점검 : grep 명령어

- 감염 단말에 설정된 악성코드 자동 실행 파일을 점검합니다.
  - 점검 대상 : /etc/sysconfig/\*
  - 점검 결과 : [ -f "악성코드 경로" ] && "악성코드 경로"
- ⚠ 오진(정상) 가능성이 있으므로 의심 파일 대상 초동 점검(8) 및 YARA 기반 점검(9) 필요

# 명령어 2-1

```
$ sudo grep -Er '\[s*-f\s+/[^\]]+\]\s*&&\s*/' /etc/sysconfig/
```

```
/etc/sysconfig/test:[ -f /usr/sbin/smartadm ] && /usr/sbin/smartadm
```

# 명령어 2-2 : 호환성 이슈로 명령어 2-1 이 실행되지 않는 경우

```
$ sudo find /etc/sysconfig/ -type f -exec egrep '\[s*-f\s+/[^\]]+\]\s*&&\s*/' {} +
```

### 3. BPF(Berkeley Packet Filter) 점검 : ss 명령어

- 시스템에 등록된 BPF를 조회하여 악성코드 실행 여부를 점검합니다.
  - 점검 가능 버전 : Linux Kernel 3.2 이상 + iproute2 4.0 이상
  - CentOS 6.x 이하 버전에서는 확인 불가 → "3. RAW 소켓 사용 확인" 점검
- 점검 스크립트 : [붙임1] BPF 점검 스크립트 : bpfdoor\_bpf.sh

# 명령어 3-1 : 서버에 등록된 BPF 필터 확인(비정상 필터 유무 점검)

```
$ sudo ss -0pb
```

# BPF 필터 내 매직넘버(Magic Sequences) 확인 → 악성코드 감염 의심

# 명령어 3-2

```
$ sudo ss -0pb | grep -E "21139|29269|960051513|36204|40783"
```

# 명령어 3-3 : 명령어 3-2로 확인 불가 시 사용

```
$ sudo ss -0pb | grep -EB1 "$((0x5293))|$((0x7255))|$((0x39393939))|$((0x8D6C))|$((0x9F4F))"
```

- 악성코드 감염 시 출력되는 BPF 값 : 붉은색 숫자 확인

```
[root@localhost ~]# ss -0pb | grep -E "21139|29269|960051513"
bpf filter (39): 0x28 0 0 12, 0x15 0 36 2048, 0x30 0 0 23, 0x15 0 5 17, 0x28 0 0 2
0, 0x45 32 0 8191, 0xb1 0 0 14, 0x48 0 0 22, 0x15 28 29 29269, 0x15 0 7 1, 0x28 0 0 20, 0x4
5 26 0 8191, 0xb1 0 0 14, 0x48 0 0 22, 0x15 0 23 29269, 0x50 0 0 14, 0x15 20 21 8, 0x15 0 2
0 6, 0x28 0 0 20, 0x45 18 0 8191, 0xb1 0 0 14, 0x50 0 0 26, 0x54 0 0 240, 0x74 0 0 2, 0x0c
0 0 0, 0x07 0 0 0, 0x48 0 0 14, 0x15 9 0 21139, 0xb1 0 0 14, 0x50 0 0 26, 0x54 0 0 240, 0x7
4 0 0 2, 0x04 0 0 26, 0x0c 0 0 0, 0x07 0 0 0, 0x40 0 0 14, 0x15 0 1 960051513, 0x06 0 0 655
35, 0x06 0 0 0,
```

### 4. RAW 소켓 사용 점검 : lsnf 명령어

- "SOCK\_RAW", "SOCK\_DGRAM" 소켓을 사용중인 프로세스를 점검합니다.
  - 오진(정상) 가능성이 있으므로 의심 파일 대상 초동 점검(6) 및 YARA 기반 점검(7) 필요
  - "IP type=SOCK\_DGRAM" 값 오진이 많은 경우 명령어 제외 후 점검

- ⚠ 명령어 유형2 는 시스템 내 inode 전체를 탐색 하므로 부하 가능성이 존재합니다.
  - (권장사항) 아래 세 가지를 만족한 경우에만 사용
    - ①뮤텍스/락 파일 존재, ②BPF 점검 결과 없음, ③ RAW 소켓 명령어 4-1 결과 없음

```
# 명령어 4-1
$ sudo lsof 2>/dev/null | grep -E "IP type=SOCK_RAW|IP type=SOCK_DGRAM" | awk '{print $2}' | sort -u | xargs -r ps -fp
```

```
UID      PID  PPID  C STIME TTY      TIME CMD
root    13851   1  0 Apr30 ?      00:00:00 dbus-daemon --system
```

```
# 명령어 4-2 : 명령어 4-1에서 진단되지 않는 경우 사용
# Linux Kernel 4.0 이상에서 점검, inode 수에 따라 시스템 부하 가능성 존재
$ sudo awk '$4=="0800" && $5=="0" {print $9}' /proc/net/packet | while read inode; do sudo grep -r "ino:\s*$inode" /proc/*/fdinfo/ 2>/dev/null | awk -F/ '{print $3}' | sort -u | xargs -r sudo ps -fp; done
```

```
UID      PID  PPID  C STIME TTY      TIME CMD
root     4162  1875  0 12:31 ?      00:00:00 /usr/sbin/smartd -n -q never
```

- 의심 프로세스의 파일 경로를 확인합니다.
  - 악성 파일이 삭제된 경우 아래와 같이 표기, 메모리에서 동작 상태
    - (예시) /dev/shm/kdmtmpflush (deleted)

```
# 명령어 4-3
$ sudo ls -l /proc/<PID>/exe

lrwxrwxrwx. 1 root root 0 May  1 00:28 /proc/13851/exe -> /dev/shm/dbus-srv_bin
```

- 문자열 기반 초동 점검(6) 또는 YARA Rule 기반 점검(7)으로 악성코드 여부를 확인합니다.

## 5. 프로세스 환경변수 점검

- 셸(Shell) 연결 시 사용하는 악성코드의 환경변수를 점검합니다.
  - 점검 스크립트 : [붙임2] 환경변수 점검 스크립트 : bpfdoor\_env.sh

```
# 명령어 5-1
$ sudo ./bpfdoor_env.sh ([붙임2] 스크립트 사용)
```

```
[*] Starting BPFdoor environment variable manipulation detection...
```

```
Target : HOME=/tmp, HISTFILE=/dev/null, MYSQL_HISTFILE=/dev/null
Warning: Process with all suspicious environment variables detected (PID: 4076)
→ root    4076    4075 qmgr -l -t fifo -u
```

- 의심 프로세스의 파일 경로를 확인합니다.

```
# 명령어 5-2
$ sudo ls -l /proc/<PID>/exe

lrwxrwxrwx. 1 root root 0 4월 28 12:31 /proc/4076/exe → /dev/shm/smartadm
```

- 문자열 기반 초동 점검(8) 또는 YARA Rule 기반 점검(9)으로 악성코드 여부를 확인합니다.

## 6. 특정 포트 확인 및 네트워크 장비를 이용한 패킷 점검

- (1) 특정 네트워크 포트(port)를 사용하는지 점검합니다.
  - port : 42391~43390, 8000
  - ⚠ 오진(정상) 가능성이 있으므로 의심 파일 대상 초동 점검(8) 및 YARA 기반 점검(9) 필요

```
# 명령어 6-1
$ sudo netstat -tulpn

$ sudo netstat -tulpn 2>/dev/null | awk '{match($0, /:([0-9]+)/, a); if ((a[1] >= 42391 && a[1] <= 43390) || $0 ~ /:8000(^[^0-9]|$)/) print $0}'

$ sudo netstat -tulpn 2>/dev/null | awk '$1=="tcp"&&($6=="LISTEN"||$6=="ESTABLISHED"){lp=substr($4,index($4,":")+1);rp=substr($5,index($5,":")+1);if((lp>=42391&&lp<=43390)||lp==8000||(rp>=42391&&rp<=43390)||rp==8000)print}'

tcp 0 0 0.0.0.0:8000 0.0.0.0:* LISTEN 80002/abrt
```

- 의심 프로세스의 파일 경로를 확인합니다.

```
# 명령어 6-2
$ sudo ls -l /proc/<PID>/exe

lrwxrwxrwx. 1 root root 0 May 7 00:35 /proc/80002/exe → /home/user/control
```

- 문자열 기반 초동 점검(8) 또는 YARA Rule 기반 점검(9)으로 악성코드 여부를 확인합니다.

- (2) 네트워크 장비를 이용한 패킷 점검



#### 주요 특징

- BPFdoor 매직패킷을 감염 단말에 전송 : 24byte(UDP, TCP) , 44byte(ICMP) , 100byte 이하 (TCP:HTTP POST)
  - 전체 패킷 크기가 아닌 페이로드 크기
  - 일부 악성코드의 경우 HTTP 요청 값 수신 : TCP
    - 마커(Marker) : HTTP POST Request-URI 내 문자열 값 9999 확인
    - 페이로드 : BODY의 Base16 디코딩 값
- 페이로드 내 매직 시퀀스(sequences), 비밀번호(password) 존재
- 매직 시퀀스 : UDP, ICMP( 0x7255 , 0x9F4F ), TCP( 0x5293 , 0x39393939 , 0x8D6C )



#### 주의

- 해당 특징은 위협 행위자에 의해 쉽게 변경 가능
  - 매직 시퀀스 단일로 Rule 적용 시 오탐 가능성 존재(내부 검토 후 커스텀하여 적용)
  - 의심 패킷 탐지 → 분석 → 검증 후 차단 필요
- ※ 패킷 캡처 : TrendMicro에서 공개한 BPF 컨트롤러 사용

- UDP 패킷

Frame 80250: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on ^

Ethernet II, Src: VMware\_0d:c0:08 (00:0c:29:0d:c0:08), Dst: VMware\_e1:98:0

Source: VMware\_0d:c0:08 (00:0c:29:0d:c0:08)

Destination: VMware\_e1:98:0a (00:0c:29:e1:98:0a)

Type: IPv4 (0x0800)

[Stream index: 39]

Internet Protocol Version 4, Src: 10.10.5.74, Dst: 10.10.5.73

User Datagram Protocol, Src Port: 33001, Dst Port: 5353

Source Port: 33001

Destination Port: 5353

Length: 32

Checksum: 0x85a2 [unverified]

[Checksum Status: Unverified]

[Stream index: 925]

[Stream Packet Number: 1]

[Timestamps]

UDP payload (24 bytes)

Multicast Domain Name System (query)

Transaction ID: 0x7255

Flags: 0x0000 Standard query

Questions: 65535

Answer RRs: 65535

Authority RRs: 9999

Additional RRs: 27253

Queries

Magic Sequences (0x7255)

BPFDoor Password (justforfun)

Wireshark packet capture showing an ICMP Echo request from 10.10.5.74 to 10.10.5.73. The packet contains a magic sequence 0x7255 (justforfun) in the data field. A red box highlights the 'Data (44 bytes)' field, and a red arrow points to the hex value 7255. A red text overlay reads 'Magic Sequences (0x7255)'. A blue text overlay reads 'BPFDoor Password (justforfun)'.

```
Acknowledgment Number: 1 (relative ack number)
Acknowledgment number (raw): 2674205154
1000 ..... = Header Length: 32 bytes (8)
> Flags: 0x018 (PSH, ACK)
Window: 229
[Calculated window size: 14656]
[Window size scaling factor: 64]
Checksum: 0xb0b3 [unverified]
[Checksum Status: Unverified]
Urgent Pointer: 0
Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
  > TCP Option - No-Operation (NOP)
  > TCP Option - No-Operation (NOP)
  > TCP Option - Timestamps: TSval 17507714, TSecr 3529203954
[Timestamps]
  [Time since first frame in this TCP stream: 0.000394000 seconds]
  [Time since previous frame in this TCP stream: 0.000075000 seconds]
[SEQ/ACK analysis]
  [RTT: 0.000323000 seconds]
  [Bytes in flight: 24]
  [Bytes sent since last PSH flag: 24]
  TCP payload (24 bytes)
SSH Protocol
  Packet Length (encrypted): 52930000
  Encrypted Packet: 0a0a54a270f6a757374666f7266756e00000000
  [Direction: client-to-server]
```

- HTTP POST Request-URI에서 마커(Marker) 값 9999 (string) 존재 확인
  - 외부 ↔ 내부, 내부 ↔ 내부 간 통신 구간에서 URI 또는 User-Agent 값 점검
- BODY 값을 Base16 디코딩하여 페이로드로 사용

```
/admin/login.aspx?id=99990
/admin_login.aspx?id=99990
/Admin/Default.aspx?=99990
/UploadFile.aspx?id=099990
/user/admin.php?id=0099991
/uploadfiles.php?id=099991
/uploadfilm.php?id=0099991
/uploadphoto.php?id=099991
/uploadPic.php?id=00099991
/systems/login.php?i=99991
```

User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.5389.90 Safari/537.36

## BPFDoor 컨트롤러 점검 방법

- 컨트롤러(Controller)는 BPFDoor에 감염된 단말을 제어하기 위해 사용하는 악성 파일입니다.

### 7. 실행중인 프로세스명 점검 : ps 명령어

- 악성코드의 위장 프로세스명을 확인합니다.
  - ⚠ 오진(정상) 가능성이 있으므로 의심 파일 대상 초동 점검(8) 및 YARA 기반 점검(9) 필요
  - 비정상 프로세스 판단 기준
    - 의심 프로세스명 : /usr/sbin/abrttd , /sbin/udevvd , cmathreshd, /sbin/sgaSolAgent , /usr/sbin/atd , pickup

```
# 명령어 7-1(의심 프로세스 동작여부 확인)
$ sudo ps -ef | grep -E '/usr/sbin/abrttd|/sbin/udevvd|cmathreshd|/sbin/sgaSolAgent|/usr/sbin/atd|pickup'
```

```
root    25469   3568  0 19:18 pts/0    00:00:00 /usr/sbin/abrttd
```

- 명령어 7-2를 실행하여 파일 경로와 프로세스명의 동일여부 확인(동일한 경우 정상, 동일하지 않은 경우 비정상 의심)

```
# 명령어 7-2(예시: 비정상 의심 프로세스)
$ sudo ls -l /proc/<PID>/exe

lrwxrwxrwx. 1 root root 0  4월 29 19:19 /proc/25469/exe -> /dev/shm/controller
```

- 문자열 기반 초동 점검(8) 또는 YARA Rule 기반 점검(9)으로 악성코드 여부를 확인합니다.

## 악성 의심 파일 점검 방법

### 8. 문자열 기반 초동 점검 : strings 명령어

- 악성코드가 사용하는 문자열 패턴을 기반으로, 의심 파일을 초기 단계에서 신속히 점검합니다.
  - YARA 도구가 설치되지 않은 환경에서 초동으로 악성 의심 판단
  - 악성 의심파일 1차 선별 → 점검(임시) 디렉토리로 복사 → strings 명령 실행
- 초동 점검에서 문자열 패턴 확인 시 악성 파일을 채증하여 YARA 도구로 2차 검증 수행
  - ⚠ 초동 점검은 오진 가능성이 있으며, 반드시 2차 YARA 검증이 필요합니다.



```
# 명령어 8-1 : 단일 파일(strings) : 권장사항
$ sudo strings -a -n 5 <의심파일 경로> | grep -E 'MYSQL_HISTFILE=/dev/null|:h:d:l:
s:b:t|:f:wiunomc|:f:x:wiuoc|ttcompat'

/dev/ptmH
ttcompat

# 명령어 8-2 : 특정 경로(find), filesize : 15K~4M, 검색 깊이 1, "권장하지 않음"
# 출력 결과 [파일명]: 매칭 문자열
$ sudo find <의심경로> -maxdepth 1 -type f -size +15k -size -4M -exec sh -c 'strin
gs -a -n5 "$1"|sed "s|^|$1: |"|grep -E "MYSQL_HISTFILE=/dev/null|:h:d:l:s:b:t|:f:wi
unomc|:f:x:wiuoc|ttcompat"' _ {} \;

./dbus-srv-bin: ttcompat
./manx: export MYSQL_HISTFILE=/dev/null
./manx: :h:d:l:s:b:t:D:g:H:f:wiunomcv
```

## 9. YARA Rule 기반 점검

- 악성코드 의심 파일을 선별하여 YARA 도구와 룰로 점검합니다.
  - 적용룰 : [붙임3] BPFDoor YARA Rule : bpfdoor.yar
- 특정 디렉토리를 대상으로 YARA 도구를 사용할 경우 시스템 장애가 발생할 수 있습니다.
  - 디렉토리 내부의 바이너리 파일을 외부 또는 다른 경로에 복사하여 점검 필요

## 침해사고 신고 안내



### 침해사고 신고

- 악성코드 감염 또는 침해가 의심되거나 확인된 경우 침해사고 신고
- KISA보호나라 누리집 : <https://boho.or.kr> (침해사고 신고 → 신고하기)

과학기술정보통신부와 한국인터넷진흥원은 아래와 같이 『정보통신망 이용 촉진 및 정보보호 등에 관한 법률』 (이하, 정보통신망법) 에 근거하여 민간분야 인터넷 침해사고 예방 및 대응 활동을 수행하고 있습니다.

☞ 제 48조의3(침해사고의 신고 등) : 정보통신서비스 제공자의 침해사고 신고 접수

※ 침해사고 신고를 아니할 시, 동법 제76조(과태료)에 근거하여 1천만원 이하의 과태료 부과

🔍 제 48조의4(침해사고의 원인 분석 등) : 정보통신서비스 제공자의 침해사고 관련 자료 보존 및 제출 요구, 현장조사 등

※ 침해사고 관련 자료 보존 명령을 위반할 시, 동법 제73조(벌칙)에 근거하여 2년 이하의 징역 또는 2천만원 이하의 벌금 부과

※ 침해사고 관련 자료를 제출하지 아니하거나 거짓으로 제출할 시, 동법 제76조(과태료)에 근거하여 1천만원 이하의 과태료 부과

정보통신서비스 제공자는 침해사고 발생 즉시 정보통신망법 48조의3(침해사고의 신고 등)에 따라 과학기술정보통신부장관이나 한국인터넷진흥원에 신고하여야 하며, 제48조의4(침해사고의 원인 분석 등)침해사고의 원인분석 및 그에 따라 피해의 확산 방지를 위하여 사고 대응, 복구 및 재발방지에 필요한 조치를 하여야 합니다.

## [붙임1] BPF 점검 스크립트 : bpfdoor\_bpf.sh

- BPF를 설정한 프로세스 정보(프로세스명, PID, 실행 경로) 출력

```
#!/bin/bash
echo "[*] Detecting processes with active BPF usage..."

# 1. Extract PIDs directly from ss output
sudo ss -Opb | grep -oP 'pid=\K[0-9]+' | sort -u | while read pid; do
    # 2. For each PID, find the executable path
    if [[ -e "/proc/$pid" ]]; then
        exe_path=$(readlink -f /proc/$pid/exe 2>/dev/null)
        proc_name=$(cat /proc/$pid/comm 2>/dev/null)
        echo ""
        echo "Process Name: ${proc_name:-Unknown}, PID: $pid"
        echo " → Executable: ${exe_path:-Not found}"
    fi
done
```

- 실행 결과

```
[root@localhost home]# ./bpfdoor_bpf.sh
[*] Detecting processes with active BPF usage...

Process Name: /usr/libexec/po, PID: 21060
→ Executable: /dev/shm/kdmtmpflush (deleted)

Process Name: sh, PID: 21061 ※ Reverse Shell
→ Executable: /usr/bin/bash 연결 시 확인됨

Process Name: avahi-daemon: c, PID: 3859
→ Executable: /dev/shm/kdmtmpflush (deleted)

Process Name: NetworkManager, PID: 873 정상 프로세스
→ Executable: /usr/sbin/NetworkManager
```

악성 의심  
(YARA 검사 필요)

## [붙임2] 환경변수 점검 스크립트 : bpfdoor\_env.sh

- 악성코드의 가상터미널(PTY), 셸의 특정 환경 변수를 사용하는 프로세스 출력

```
#!/bin/bash

echo "echo [*] Detecting processes with BPFDoor environment variable manipulation..."
echo "Target : HOME=/tmp, HISTFILE=/dev/null, MYSQL_HISTFILE=/dev/null"

CHECK_ENV=("HOME=/tmp" "HISTFILE=/dev/null" "MYSQL_HISTFILE=/dev/null")

# Process scanning
for pid in $(ls /proc/ | grep -E '[0-9]+$'); do
    if [ -r /proc/$pid/environ ]; then
        env_data=$(tr '\0' '\n' < /proc/$pid/environ)

        match_all=true
        for check_item in "${CHECK_ENV[@]}; do
            if ! echo "$env_data" | grep -q "$check_item"; then
                match_all=false
                break
            fi
        done

        if [ "$match_all" = true ]; then
            echo "Warning: Process with all suspicious environment variables detected (PID: $pid)"
            echo "    → $(ps -p $pid -o user=,pid=,ppid=,cmd=)"
            echo ""
        fi
    fi
done
```

- 실행 결과

```
[root@localhost /]# ./bpfdoor_env.sh
echo [*] Detecting processes with BPFDoor environment variable manipulation...
Target : HOME=/tmp, HISTFILE=/dev/null, MYSQL_HISTFILE=/dev/null
Warning: Process with all suspicious environment variables detected (PID: 21061)
→ root      21061    21060 qmgr -l -t fifo -u
```

### [붙임3] BPFDoor YARA Rule : bpfdoor.yar

- BPFDoor 탐지 : ELF\_BPFDoor
- BPFDoor 컨트롤러 탐지 : ELF\_BPFDoor\_Controller

```
rule ELF_BPFDoor
{
    meta:
        description = "BPFDoor Malware(ELF) Detection rule"
        author = "KrCERT/CC Threat Hunting Analysis Team"
        date = "2025-05-06"
        hash = "c7f693f7f85b01a8c0e561bd369845f40bff423b0743c7aa0f4c323d9133b5d4"
        hash = "3f6f108db37d18519f47c5e4182e5e33cc795564f286ae770aa03372133d15c4"
        hash = "95fd8a70c4b18a9a669fec6eb82dac0ba6a9236ac42a5ecde270330b66f51595"
        hash = "aa779e83ff5271d3f2d270eaed16751a109eb722fca61465d86317e03bbf49e4"
        ver = "1.1"

    strings:
        $pty_1 = "/dev/ptm"
        $pty_2 = "ptem"
        $pty_3 = "ldterm"
        $pty_4 = "ttcompat"

        // for ( i = 0xA597; i <= 0xA97E; ++i )
        $bind_port_1 = {C7 45 ?? 97 A5 00 00 EB}
        $bind_port_2_1 = {83 45 ?? 01 81 7D ?? 7E A9 00 00 7E}
        $bind_port_2_2 = {81 7D ?? 7E A9 00 00 7E}
```

```

// v1.1 : added new pattern "PS1=[\\u@\\h : "
$ps1 = {C6 85 ?? ?? FF FF 50 C6 85 ?? ?? FF FF 53 C6 85 ?? ?? FF FF 31 C6 8
5 ?? ?? FF FF 3D C6 85 ?? ?? FF FF 5B C6 85 ?? ?? FF FF 5C C6 85 ?? ?? FF FF 75
C6 85 ?? ?? FF FF 40 C6 85 ?? ?? FF FF 5C C6 85 ?? ?? FF FF 68}

// v1.1 : bpf filter modify (dbus, hald, trend-B, Trend-E)
$bpf_filter_1 = {B1 00 00 00 0E 00 00 00 48 00 00 00 16 00 00 00 15 00 ?? ??
?? ?? 00 00 15 00 00 07 01 00 00 00 28 00 00 00 14 00 00 00 45 00 (11|1A) 00 FF
1F 00 00}
$bpf_filter_2 = {07 00 00 00 00 00 00 00 (40|48) 00 00 00 00 00 00 00 02 0
0 00 00 (00|01|03) 01 00 00 00 00 00 00 00 ?? ?? ?? ?? 02 00 00 00 (01|02|04) 0
0 00 00 61 00 00 00 (01|02|04) 00 00 00}
$bpf_filter_3 = {07 00 00 00 00 00 00 00 40 00 00 00 0E 00 00 00 15 00 00
01 39 39 39 39 06 00 00 00 FF FF 00 00 06 00 00 00 00 00 00 00}
$bpf_filter_4 = {48 00 00 00 10 00 00 00 15 00 01 00 BB 01 00 00 15 00 00 01
16 00 00 00 06 00 00 00 00 00 04 00 06 00 00 00 00 00 00 00}

// I5*AYbs@LdaWbsO
$md5_salt = {C6 45 ?? 49 C6 45 ?? 35 C6 45 ?? 2A C6 45 ?? 41 C6 45 ?? 59
C6 45 ?? 62 C6 45 ?? 73 C6 45 ?? 40 C6 45 ?? 4C C6 45 ?? 64 C6 45 ?? 61 C6 45
?? 57 C6 45 ?? 62 C6 45 ?? 73 C6 45 ?? 4F}

// qmgr -l -t fifo
$name = {C6 ?? ?? ?? FF FF 71 C6 ?? ?? ?? FF FF 6D C6 ?? ?? ?? FF FF 67 C6
?? ?? ?? FF FF 72 C6 ?? ?? ?? FF FF 20 C6 ?? ?? ?? FF FF 2D C6 ?? ?? ?? FF FF 6C
C6 ?? ?? ?? FF FF 20 C6 ?? ?? ?? FF FF 2D C6 ?? ?? ?? FF FF 74 C6 ?? ?? ?? FF FF
20 C6 ?? ?? ?? FF FF 66 C6 ?? ?? ?? FF FF 69 C6 ?? ?? ?? FF FF 66 C6 ?? ?? ?? FF
FF 6F}

$solaris_1 = "(udp[8:2]=0x7255) or (icmp[8:2]=0x7255)"
$solaris_2 = "HISTFILE=/dev/null"
$solaris_3 = "MYSQL_HISTFILE=/dev/null"

condition:
uint32(0) == 0x464C457F and
(
    (all of ($pty*) and (2 of ($bind_port*) or $ps1)) or // v1.1
    ((all of ($pty*) or $ps1) and 2 of ($bind_port*)) or // v1.1
    (1 of ($bpf_filter*)) or
    ($md5_salt or $name) or
    (all of ($solaris*))
)
}

```

```

rule ELF_BPFDoor_Controller
{
    meta:
        description = "BPFDoor Malware(ELF) Controller Detection rule"
        author = "KrCERT/CC Threat Hunting Analysis Team"
        date = "2025-05-06"
        hash = "93f4262fce8c6b4f8e239c35a0679fbbbb722141b95a5f2af53a2bcaf
e4edd1c"
        hash = "1925e3cd8a1b0bba0d297830636cdb9ebf002698c8fa71e006358120
4f4e8345"
        hash = "591198c234416c6ccbcea6967963ca2ca0f17050be7eed1602198308
d9127c78"
        ver = "1.1"

    strings:
        // v1.1 : added new pattern
        $pname = /\usr\sbin\[a-z]{2,8}/
        $str_1 = "[-] option requires an argument -- d"
        $str_2 = ":h:d:l:s:b:t:"

        // mov cs:magic_flag, 5571h
        $magic = {C7 05 ?? ?? ?? 00 71 55 00 00}

        $env_1 = "export TERM=vt100"
        $env_2 = "export MYSQL_HISTFILE=/dev/null"
        $env_3 = "export HISTFILE=/dev/null"
        $env_4 = "unset PROMPT_COMMAND"
        $env_5 = "export HISTSIZE=100"

    condition:
        uint32(0) == 0x464C457F and
        ($pname or 1 of ($str*)) and $magic and (all of ($env*))
}

```