K-Shield Jr.

BPFDoor 변종 분석 보고서

2025.06

손정호게임 분석팀

이강호, 유정은

목 차

1. 배경

BPFDoor는 리눅스 기반 백도어 악성코드로 APT 공격 그룹의 정찰 및 장기적인 접근 통제 수단으로 활용되고 있다. 특히 이 악성코드는 BPF(Berkeley Packet Filter)를 이용한 패킷 감시 방식으로 작동하며 기존의 포트 기반 네트워크 탐지나 프로세스 기반 보안 솔루션으로는 탐지가 매우 어려운 특징을 가진다. 일반적인 리버스 쉘 백도어나 C2 통신 방식과 달리 BPFDoor는 자체적으로 네트워크 패킷을 수신 대기하면서 사전 정의된 패턴을 감지할 경우에만 동작을 수행하는 방식으로 설계되어 있어 디스크 흔적 없이도 오랜 시간 은밀히 시스템에 상주하여 동작할 수 있다.

BPFDoor의 가장 큰 특징 중 하나는 지속적으로 변종이 제작·유포되고 있다는 점이다. 현재까지 발견된 BPFDoor 악성코드들은 코드 구성과 동작 방식에 따라 대체로 다섯 가지 큰 유형(A~E)으로 분류되며 이들 간에는 패킷 필터 구성, 은폐 기법 등에서 중요한 차이를 보인다. 그럼에도 불구하고 이들 유형은 공통적으로 BPF를 이용한 비정상적 트래픽 감시 및 바인드 & 리버스 셸 생성이라는 구조를 공유한다는 점에서 상호 간의 유사성을 기반으로 한 비교 분석이 필요하다.

이러한 변종 분석을 통해 BPF 기반 악성코드에 대해 깊이 있게 이해할 수 있으며 이는 결과적으로 보다 정밀하고 효과적인 BPF 기반 악성 행위 탐지 스크립트의 개발로 이어질 수 있다. 또한 본 보고서에서 제시하는 비교 분석 결과는 향후 BPFDoor의 신규 변종 탐지 및 유사한 BPF 기반 악성코드 대응 전략 수립에 있어서도 중요한 레퍼런스로 활용할 수 있을 것이다.

2. 분석 대상 및 환경 정보

SKT 해킹 사고와 관련하여 한국인터넷진흥원(KISA)는 최근 해킹공격에 악용된 BPFDoor 계열 악성코드 4종의 정보를 공개하였다[1]. 공개된 4종은 hpasmmld, smartadm, hald-addon-volume, dbus-srv-bin.txt 파일로 이를 기반으로 본 분석에서는 [표 1]에서 제시하는 환경에서 정적 및 동적 분석을 진행하였다.

VM workstation	17.0.0
OS	Ubuntu 24.04.2 LTS
Kernel	Linux 6.11.0-26-generic
System Architecture	x86_64
IDA Pro	9.0

표 1. 실행 및 분석 환경

3. 사용된 BPFDoor 변종 정적 분석

IDA Pro 9.0을 통해 각 실행파일을 디컴파일하여 C코드 기반으로 분석하였으며 그 외의 경우에는 리눅스에서 strings를 통해 문자열 기반 분석을 진행하였다. 이를 기반으로 4개의 파일을 유사점 및 차이점으로 분석한다.

3.1 유사점

1) 자가 복제 및 삭제 기능 제거

```
_BOOL8 __fastcall sub_403024(__int64 a1, __int64 a2)
{
  char v3[112]; // [rsp+30h] [rbp-170h] BYREF
  _BYTE v4[256]; // [rsp+A0h] [rbp-100h] BYREF

  memset(v4, 0, sizeof(v4));
  strcpy(v3, "/bin/rm -f /var/lock/%s;/bin/cp %s /var/lock/%s && /bin/chmod 755 /var/lock/%s && /var/lock/%s --init");
  sub_4021D0(v4, 256LL, v3, a2, a1, a2, a2, a2, a2);
  sub_402180(v4);
  sub_402680(2LL);
  return (unsigned int)sub_402570(qword_808190, 4LL) != 0;
}
```

그림 1. 복제 및 삭제되지 않는 프로세스

/bin/rm -f /var/lock/%s; /bin/cp %s /var/lock/%s && /bin/chmod 755 /var/lock/%s && /var/lock/%s --init 명령을 실행한다. /var/lock 폴더는 리소스 잠금을 위해 사용하는 폴더로

잠금 상태를 기록하는 역할을 한다. 따라서 이는 기존의 BPFDoor와 비교하여 자가 복제 및 삭제 기능이 제거되었음을 확인할 수 있다.

2) 패스워드 생성 방식

```
void __noreturn sub_4011E0()
{
  char v0[16]; // [rsp+10h] [rbp-70h] BYREF
  _QWORD v1[10]; // [rsp+20h] [rbp-60h] BYREF
  char v2; // [rsp+70h] [rbp-10h]
  int v3; // [rsp+7Ch] [rbp-4h]

  v3 = 0;
  memset(&v1[6], 0, 32);
  v2 = 0;
  memset(v1, 0, 40);
  strcpy(v0, "I5*AYbs@LdaWbsO");
  sub_400430(v1, v0);
}
```

그림 2. I5*AYbs@LdaWbsO salt 값

기존 BPFDoor에서 명령 인증을 위한 비밀번호 방식이 salt 상수 I5*AYbs@LdaWbsO 을 이용하는 것으로 대체되었다. 따라서, salt 값을 활용하여 MD5 알고리즘을 적용하게 된다.

3) SSL 암호화 통신

```
v49 = "-----BEGIN CERTIFICATE-----\n"
      "MIIB+zCCAWQCCQCtA0agZ+qO5jANBgkqhkiG9w0BAQsFADBCMQswCQYDVQQGEwJY\n"
      "WDEVMBMGA1UEBwwMRGVmYXVsdCBDaXR5MRwwGgYDVQQKDBNEZWZhdWx0IENvbXBh\n"
      "bnkgTHRkMB4XDTIxMTEyMzAyMTc0NloXDTMxMTEyMTAyMTc0NlowQjELMAkGA1UE\n"
      "BhMCWFgxFTATBgNVBAcMDERlZmF1bHQgQ2l0eTEcMBoGA1UECgwTRGVmYXVsdCBD\n"
      "b21wYW55IEx0ZDCBnzANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEAx1JvO+nqr24g\n"
      "wc8at6x1NtZt7DoDi1/Ge/F70zz4gbxX/OxhOxXKexYrphsHXBzYVEWOyof9Vnok\n"
      "ST7GKdRiRg6OS90WfdWFoVN2EdxwBN+BdozmwRBG1DAdqAhbeUcUFeZO0Fbuo7fr\n"
      "FvTfsC31khj6ioKJl0d4kfo2zLk6WhcCAwEAATANBgkqhkiG9w0BAQsFAAOBgQA1\n"
      "iC/5g+eN3Hq/627tMbLhipNUtC0OEdtpq20mbUIMXTRYh4kZPAah1LZqx2h72BV1\n"
      "i8pYJo34kZ/3HyV6UJtBf/jJv1fprEWvo2Lj8YrCpagXh82i7353GUeiKFVr0gx+\n"
      "4ruTus1m0bX1NZN6XRAbgzar7bfki0HHjWxJB8NRLQ==\n"
      "-----END CERTIFICATE-----";
v48 = "-----BEGIN RSA PRIVATE KEY-----\n"
      "MIICXAIBAAKBgQDHUm876eqvbiDBzxq3rHU21m3sOgOLX8Z78XvTPPiBvFf87GE7\n"
      "Fcp7FiumGwdcHNhURY7Kh/1WeiRJPsYp1GJGDo5L3RZ91YWhU3YR3HAE34F2jObB\n"
      "EEbUMB2oCFt5RxQV5k7QVu6jt+sW9N+wLfWSGPqKgomXR3iR+jbMuTpaFwIDAQAB\n"
      "AoGAJcrBgHCnqL+OwnCMNksiplUd3m5ZgbGVJqbwvWqQC7k0TaZcASHulEvt1r7F\n"
      "NHfbpI7Tth72r9DU1HZsiD3Wq++dNCn3ep+6/lKPldzhsrz+XAQiNWZkB6JLNxB4\n"
      "Ty6yeg2Z0IXnceAf0jNdbK7fqCNhvVIVeYyOBpy0On+BTJECQQD879Tyd1R07+Fs\n"
      "J6cGw37oEktQbd07SFvQ0ZN87BWgjmIh41CxYuJ3us8sVBYpFw2upQ77b2+YZlbM\n"
      "JwUgA44FAkEAybxiFZrSs1SQXQcgNGkpp84/KzQBsUgu+zyPQabbXkx+5qnBVN6J\n"
      "wgdGmvC3D8f/wi7ms+poP/P2pIv8GQhmawJAT/u3JwU9G81PR1gypRzk4JYIYuKa\n"
      "9sgm4J2lOfedzyu3NGgheDaAzsOwB6B/cXbsyLFBT5kpfNO/1sbwA/Ob0QJAQqPR\n"
      "PtPLAGRR3/knoUiP8xfFdC8U1AxFQMbrh8NnucnJLccrzy5IEWk34Jzdy//EM9As\n"
      "c4hecanUctyvJVHKCwJBANGpvzmXs4gL1bCQwN0LBULPI+yw5mhrCfDtBRw2zn2I\n"
      "xiWAb+d3on6kaCT51XzZjGf1srf0K8b3IpOQ4RaNBtg=\n"
      "-----END RSA PRIVATE KEY-----";
```

그림 3. 하드코딩된 인증서 및 개인키

암호화 방식은 RC4에서 RC4-MD5를 사용하는 SSL로 변경되었으며 하드코딩된 인증서와 개인키를 확인할 수 있다. base64로 인코딩된 인증서는 [그림 4]와 같은 X.509 인증서이다.

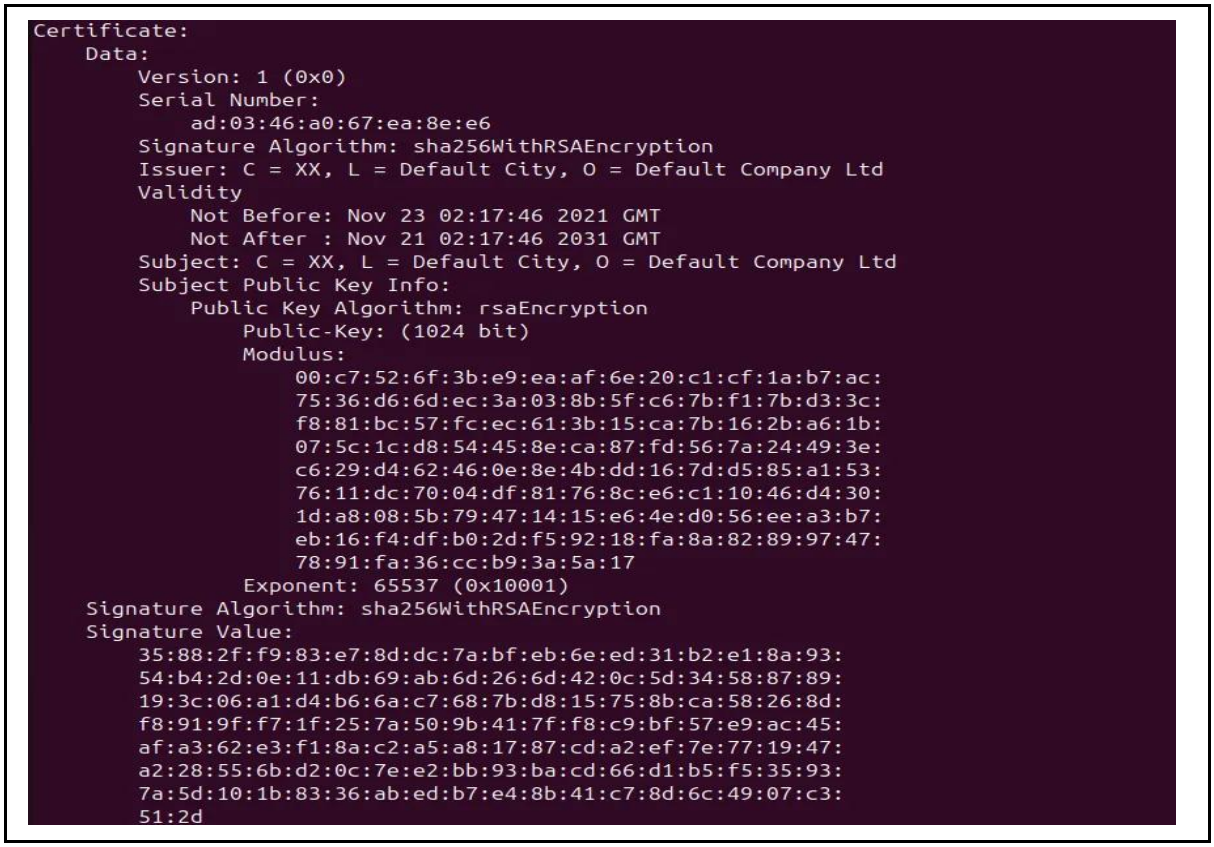

그림 4. 디코딩한 X.509 인증서

3.2 차이점

- 위장용 프로세스 이름

1) hpasmmld

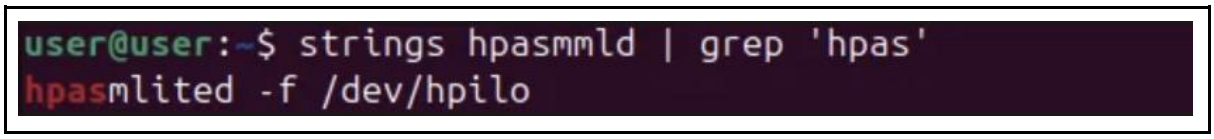

그림 5. hpasmmld 위장 프로세스

2) smartadm

```
__int64 __fastcall sub_4057A5(int a1, __int64 *a2)
{
  int v2; // eax
  int v4; // eax
  unsigned int v5; // eax
  char v6[48]; // [rsp+10h] [rbp-60h] BYREF
  char v7[40]; // [rsp+40h] [rbp-30h] BYREF
  const char *v8; // [rsp+68h] [rbp-8h]

  strcpy(v7, "73b9989bb8dd522b8e172f2e985810eb");
  strcpy(v6, "d46bf5d43cffd7793665d40fc767ed86");
  v8 = "/usr/sbin/smartd -n -q never";
```

그림 6. smartadm 위장 프로세스

3) hald-addon-volume

```
v8 = "/usr/libexec/hald-addon-volume";
```

그림 7. hald-addon-volume 위장 프로세스

4) dbus-srv-bin.txt

```
dbus-daemon --system
```

그림 8. strings 명령어를 통한 dbus-srv-bin.txt 출력값

공개된 오픈소스 BPFDoor와 마찬가지로 악성코드의 프로세스명을 위장하기 위한 프로세스 이름이 각 프로세스별로 저장되어 있다.

hpasmmld	hpasmlited -f /dev/hpilo
smartadm	/usr/sbin/smartd -n -q never
hald-addon-volume	/usr/libexec/hald-addon-volume
dbus-srv-bin.txt	dbus-daemon –system

표 2. 변종별 위장 프로세스 이름

프로세스명	의미
hpasmlited -f /dev/hpilo	HP 서버의 하드웨어 상태를 모니터링하기 위한 백그라운드 데몬을 /dev/hpilo 지정해 모니터링
/usr/sbin/smartd -n -q never	디스크 상태를 모니터링하는 데몬 프로세스인 smartd를 no check, quite 모드로 설정
/usr/libexec/hald-addon-volume	개별 저장 장치(또는 파티션)를 감시하여 해당 장치에 이벤트가 발생하면 이를 HAL에 전달
dbus-daemon --system	D-Bus(메시지 버스)를 시스템 전역에 실행

표 3. 위장 프로세스별 의미

위 표와 같이 악성코드 프로세스의 위장과 시스템 권한 획득 및 유지 목적으로 변종별 설정된 이름으로 악성코드가 위장하여 프로세스를 실행하게 된다.

- **mutex lock을 위한 프로세스 이름**

1) hpasmmld

```
__int64 sub_4031E0()
{
  qword_83B6A8 = (__int64)"/var/run/hp-health.pid";
  if ( !(unsigned int)sub_53CD20("/var/run/hp-health.pid", 4) )
    sub_4FEDE0(0);
  if ( !(unsigned int)sub_53C560() )
    sub_4003D0(&unk_83B6C0, 586LL);
  return 0LL;
}
```

그림 9. hpasmmld mutex lock 이름

2) smartadm

```
qword_808190 = (__int64)"/var/run/hald-smartd.pid";
if ( !(unsigned int)sub_402570("/var/run/hald-smartd.pid", 4LL) )
  sub_402630();
sub_402170();
```

그림 10. smartadm mutex lock 이름

3) hald-addon-volume

```
qword_809198 = (__int64)"/var/run/hald-addon.pid";
if ( !(unsigned int)sub_4025B0("/var/run/hald-addon.pid", 4LL) )
  sub_402680();
sub_4021B0();
```

그림 11. hald-addon-volume mutex lock 이름

4) dbus-srv-bin.txt

```
/var/run/system.pid
libssl.so.10
```

그림 12. dbus-srv-bin.txt mutex lock

hpasmmld	/var/run/hp-health.pid
smartadm	/var/run/hald-smartd.pid
hald-addon-volume	/var/run/hald-addon.pid
dbus-srv-bin.txt	/var/run/system.pid

표 4. mutex lock을 위한 프로세스별 pid명

중복 실행 방지 및 실행 권한 확인 역할 용도로 사용되는 pid명은 실행 시 해당 파일이 읽기 권한이 존재할 경우 이미 프로세스가 동작 중인 것으로 파악하고 exit(0);으로 종료하여 중복 실행을 방지한다. pid명 또한 기존의 A 유형 오픈소스와 다른 pid명을 사용하고 있는 것을 확인해 볼 수 있다.

3.3 기타

```
__int64 __fastcall sub_4057A5(int a1, __int64 *a2)
{
  int v2; // eax
  int v4; // eax
  unsigned int v5; // eax
  char v6[48]; // [rsp+10h] [rbp-60h] BYREF
  char v7[40]; // [rsp+40h] [rbp-30h] BYREF
  const char *v8; // [rsp+68h] [rbp-8h]

  strcpy(v7, "73b9989bb8dd522b8e172f2e985810eb");
  strcpy(v6, "d46bf5d43cffd7793665d40fc767ed86");
```

그림 13. hpasmmld를 제외한 악성코드의 패스워드 해시 값

hpasmmld의 같은 경우 다른 변종 악성코드에 있는 하드코딩된 패스워드 해시 값이 확인되지 않는다.

```
else
{
  v17 = 0;
  v10[0] = 0LL;
  v10[1] = 0LL;
  v11 = 0;
  strcpy(v9, "/usr/libexec/postfix/master");
  sub_4026E0();
  if ( v4 )
    sub_402680();
  sub_4022C0();
  sub_402380(1LL, 0LL);
  v5 = sub_402180(qword_805358);
  sub_402270(qword_805358, 0LL, v5);
  sub_4020C0(qword_805358, v9);
  sub_402410(15LL, v9);
  v6 = sub_403569((__int64)(v28 + 10));
  v17 = v6;
```

그림 14. hpasmmld를 제외한 악성코드의 /postfix 사용

또한 fork() 호출로 새로운 프로세스를 생성하고 부모 프로세스는 종료함으로써 자식 프로세스는 독립적인 데몬 프로세스가 되도록 하여 일반적인 서비스 프로세스와 유사하도록 한다. 이 과정에서 hpasmmld는 다른 변종들과 달리 /usr/libexec/postfix/master를 사용하지 않는다.

4. hpasmmld 악성코드 (A 유형 변종)

다음은 SKT 침해사고에서 사용된 BPFDoor A 유형의 변종 중 하나인 hpasmmld 악성코드 실행 후 감염된 시스템에 남는 흔적 분석을 통한 악성코드 행위에 대한 분석이다.

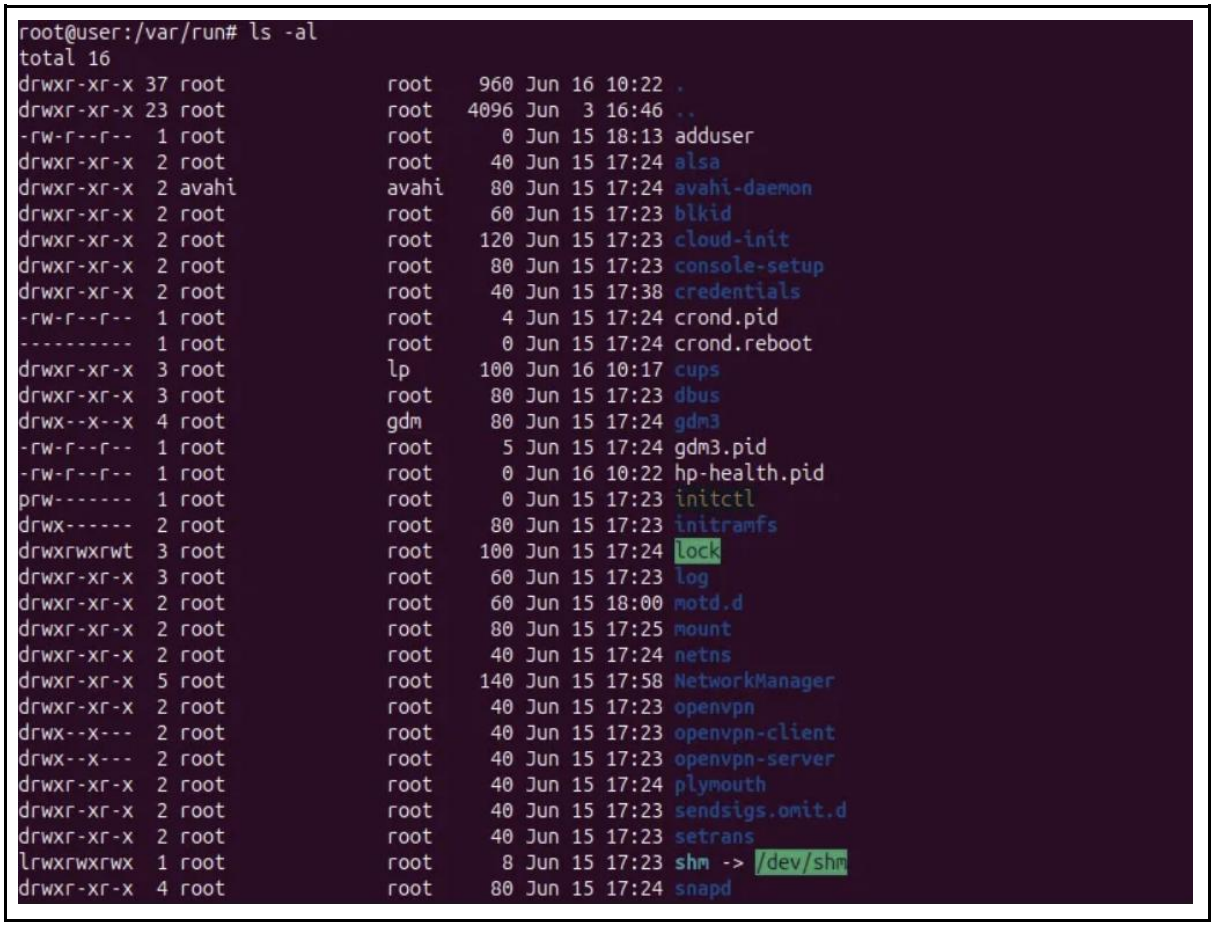

그림 15. hpasmmld mutex lock pid 명

/var/run/hp-health.pid 파일이 생성되어 중복 실행을 막고 있으며 기존 BPFDoor와 비교했을 때 /dev/shm이 삭제되지 않는다.

그림 16. hpasmmld 위장 프로세스명

pid 11010번 프로세스가 hpasmlited -f /dev/hpilo로 프로세스명이 변경되었다.

그림 17. hpasmmld 환경변수

기존의 환경변수를 삭제하여 프로세스명이 변경된 흔적을 남기지 않게한다.

그림 18. hpasmmld 프로세스 stack

/proc/[pid]에서 stack 확인을 통해 패킷 수신 처리를 위한 함수가 들어있음을 확인할 수 있으며 해당 프로세스가 패킷 수신 대기 중에 있음을 알 수 있다.

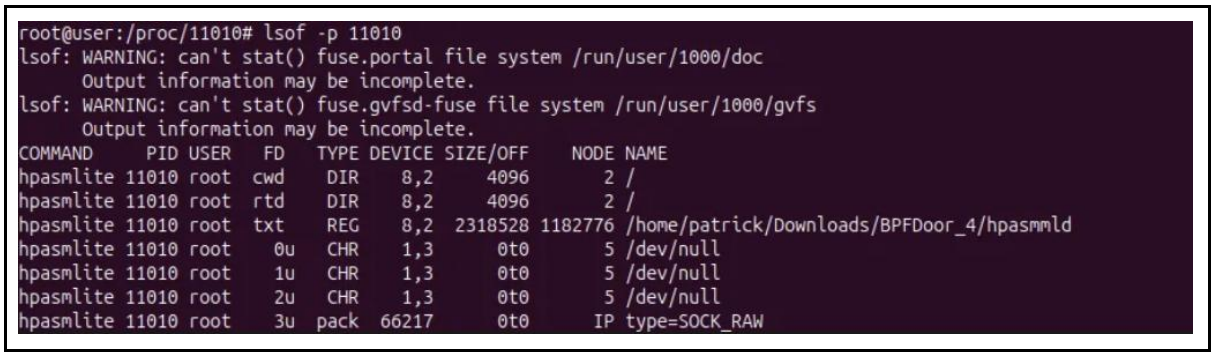

그림 19. hpasmmld 프로세스 lsof

raw socket을 사용하고 있으며 자가 복제 및 삭제 기능이 없기 때문에 삭제 흔적이 남지 않는다.

그림 20. hpasmmld 프로세스 파일 디스크립터

그림 21. hpasmmld 프로세스 소켓

파일 디스크립터로 66217번 소켓을 사용하고 있음을 파악할 수 있으며 type이 3으로 raw socket이 등록되어 있다.

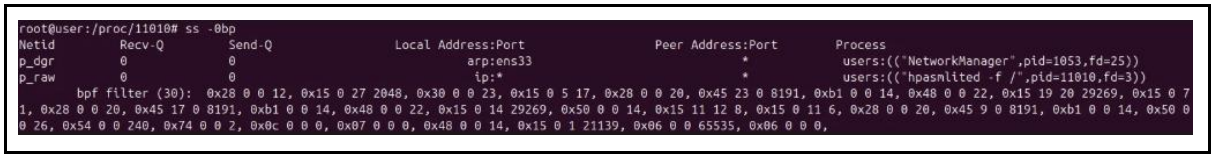

그림 22. hpasmmld 매직 필터 설정 값

ss -0bp로 30개의 명령어로 이루어진 bpf 필터를 확인할 수 있다. 이를 통해, BPFDoor A유형의 변종임을 알 수 있다.

5. hald-addon-volume (D 유형 변종)

다음은 SKT 침해사고에서 사용된 BPFDoor D 유형 변종인 hald-addon-volume 악성코드 실행 후 감염된 시스템에 남는 흔적 분석을 통한 악성코드 행위에 대한 분석이다.

```
root@user:/var/run# ls -al
total 16
drwxr-xr-x 38 root              root    980 Jun 16 10:55 .
drwxr-xr-x 23 root              root   4096 Jun  3 16:46 ..
-rw-r--r--  1 root              root      0 Jun 15 18:13 adduser
drwxr-xr-x  2 root              root     40 Jun 15 17:24 alsa
drwxr-xr-x  2 avahi             avahi    80 Jun 15 17:24 avahi-daemon
drwxr-xr-x  2 root              root     60 Jun 15 17:23 blkid
drwxr-xr-x  2 root              root    120 Jun 15 17:23 cloud-init
drwxr-xr-x  2 root              root     80 Jun 15 17:23 console-setup
drwxr-xr-x  2 root              root     40 Jun 15 17:38 credentials
-rw-r--r--  1 root              root      4 Jun 15 17:24 crond.pid
----------  1 root              root      0 Jun 15 17:24 crond.reboot
drwxr-xr-x  3 root              lp      100 Jun 16 10:48 cups
drwxr-xr-x  3 root              root     80 Jun 15 17:23 dbus
drwx--x--x  4 root              gdm      80 Jun 15 17:24 gdm3
-rw-r--r--  1 root              root      5 Jun 15 17:24 gdm3.pid
-rw-r--r--  1 root              root      0 Jun 16 10:49 hald-addon.pid
prw-------  1 root              root      0 Jun 15 17:23 initctl
drwx------  2 root              root     80 Jun 15 17:23 initramfs
drwxrwxrwt  3 root              root    100 Jun 15 17:24 lock
drwxr-xr-x  3 root              root     60 Jun 15 17:23 log
drwxr-xr-x  2 root              root     60 Jun 15 18:00 motd.d
drwxr-xr-x  2 root              root     80 Jun 15 17:25 mount
drwxr-xr-x  2 root              root     40 Jun 15 17:24 netns
drwxr-xr-x  5 root              root    140 Jun 15 17:58 NetworkManager
drwxr-xr-x  2 root              root     40 Jun 15 17:23 openvpn
drwx--x---  2 root              root     40 Jun 15 17:23 openvpn-client
drwx--x---  2 root              root     40 Jun 15 17:23 openvpn-server
drwxr-xr-x  2 root              root     40 Jun 15 17:24 plymouth
drwxr-xr-x  2 root              root     40 Jun 15 17:23 sendsigs.omit.d
drwxr-xr-x  2 root              root     40 Jun 15 17:23 setrans
lrwxrwxrwx  1 root              root      8 Jun 15 17:23 shm -> /dev/shm
drwxr-xr-x  4 root              root     80 Jun 15 17:24 snapd
```

그림 23. hald-addon-volume mutex lock pid 명

mutex lock을 위해 크기가 0인 hald-addon.pid 파일이 생성되었으며 /dev/shm 역시 삭제되지 않는다.

```
root        9145  0.7  0.3  46872 12020 ?        Ss   10:48   0:00 /usr/sbin/cupsd -l
cups-br+    9146  1.1  0.5 268344 19968 ?        Ssl  10:48   0:00 /usr/sbin/cups-browsed
root        9152  0.2  0.2  34052  9752 ?        S    10:48   0:00 /usr/lib/apt/methods/https
_apt        9157  0.1  0.2  34040  9608 ?        S    10:48   0:00 /usr/lib/apt/methods/https
root        9171  0.0  0.0  29380  1684 ?        Ssl  10:49   0:00 /usr/libexec/hald-addon-volume
root        9177  514  0.1  22412  4436 pts/1    R+   10:49   0:00 ps aux
```

그림 24. hald-addon-volume 위장 프로세스명

/usr/libexec/hald-addon-volume 프로세스명으로 위장한다.

```
root@user:/proc/9171# strings environ
don-volume
```

그림 25. hald-addon-volume 환경변수

기존 프로세스의 환경변수를 삭제함으로써 프로세스명만 변경된 흔적을 남기지 않도록 한다.

```
root@user:/proc/9171# strings stack
[<0>] futex_wait_queue+0x66/0xa0
[<0>] __futex_wait+0x154/0x1d0
[<0>] futex_wait+0x73/0x130
[<0>] do_futex+0x105/0x260
[<0>] __x64_sys_futex+0x12a/0x200
[<0>] x64_sys_call+0x1dd3/0x25f0
[<0>] do_syscall_64+0x7e/0x170
[<0>] entry_SYSCALL_64_after_hwframe+0x76/0x7e
```

그림 26. hald-addon-volume 프로세스 stack

스택 확인을 통해 mutex을 획득하지 못하고 대기 상태 있음을 확인할 수 있다.

```
root@user:/proc/9171# lsof -p 9171
lsof: WARNING: can't stat() fuse.portal file system /run/user/1000/doc
      Output information may be incomplete.
lsof: WARNING: can't stat() fuse.gvfsd-fuse file system /run/user/1000/gvfs
      Output information may be incomplete.
COMMAND    PID USER   FD   TYPE DEVICE SIZE/OFF    NODE NAME
/usr/libe 9171 root  cwd    DIR    8,2     4096       2 /
/usr/libe 9171 root  rtd    DIR    8,2     4096       2 /
/usr/libe 9171 root  txt    REG    8,2  2120632 1182775 /home/patrick/Downloads/BPFDoor_4/hald-addon-volume
/usr/libe 9171 root  mem    REG    8,2  2125328  666838 /usr/lib/x86_64-linux-gnu/libc.so.6
/usr/libe 9171 root  mem    REG    8,2    14408  666897 /usr/lib/x86_64-linux-gnu/libdl.so.2
/usr/libe 9171 root  mem    REG    8,2    14408  667295 /usr/lib/x86_64-linux-gnu/libpthread.so.0
/usr/libe 9171 root  mem    REG    8,2   236616  666656 /usr/lib/x86_64-linux-gnu/ld-linux-x86-64.so.2
/usr/libe 9171 root    0u   CHR    1,3      0t0       5 /dev/null
/usr/libe 9171 root    1u   CHR    1,3      0t0       5 /dev/null
/usr/libe 9171 root    2u   CHR    1,3      0t0       5 /dev/null
/usr/libe 9171 root    3u   raw             0t0   62784 00000000:0006->00000000:0000 st=07
/usr/libe 9171 root    4u   raw             0t0   62320 00000000:0011->00000000:0000 st=07
/usr/libe 9171 root    5u   raw             0t0   62321 00000000:0001->00000000:0000 st=07
```

그림 27. hald-addon-volume 프로세스 lsof

raw socket이 62784, 62320, 62321번으로 열려있으며 자가 삭제 기능이 없기 때문에 삭제 기록이 남지 않는다.

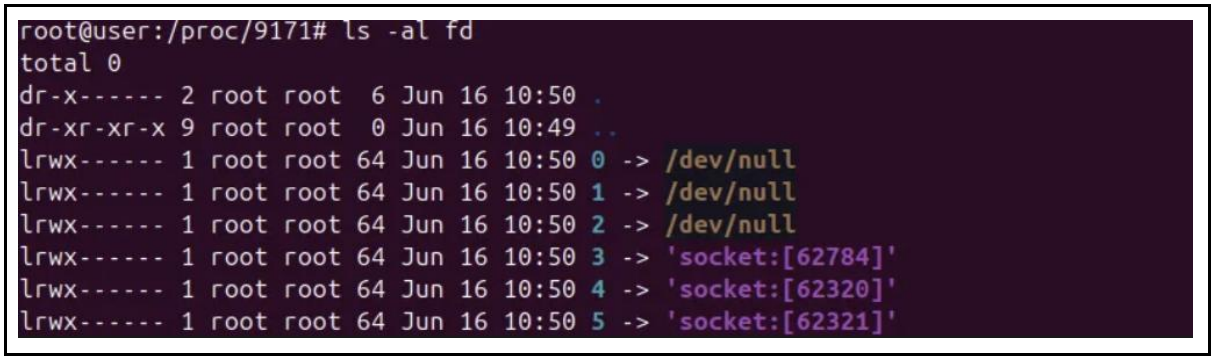

그림 28. hald-addon-volume 프로세스 파일 디스크립터

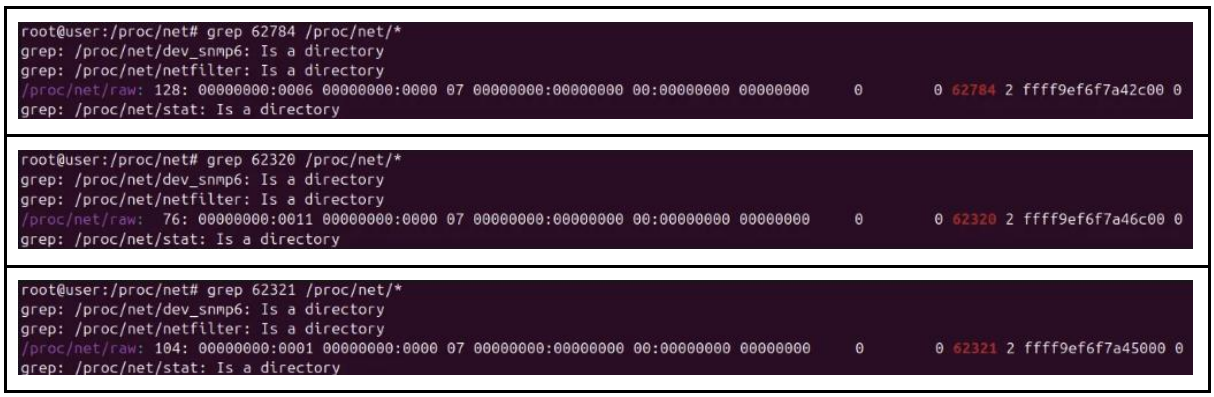

그림 29. hald-addon-volume 프로세스 소켓

파일 디스크립터로 소켓 62784, 62320, 62321번이 열려있다는 것을 볼 수 있다.

그림 30. hald-addon-volume ss -0bp 출력값

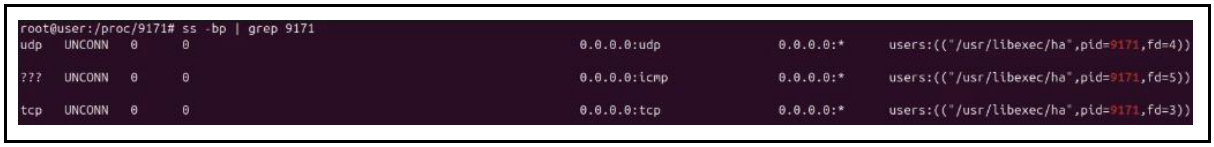

그림 31. hald-addon-volume ss -bp 출력값

ss -0bp를 통해 bpf 필터가 부착된 것은 확인되지 않았지만 ss -bp를 통해 프로세스 9171번이 각 프로토콜별(udp, icmp, tcp)로 raw socket이 열린 것을 확인할 수 있었다.

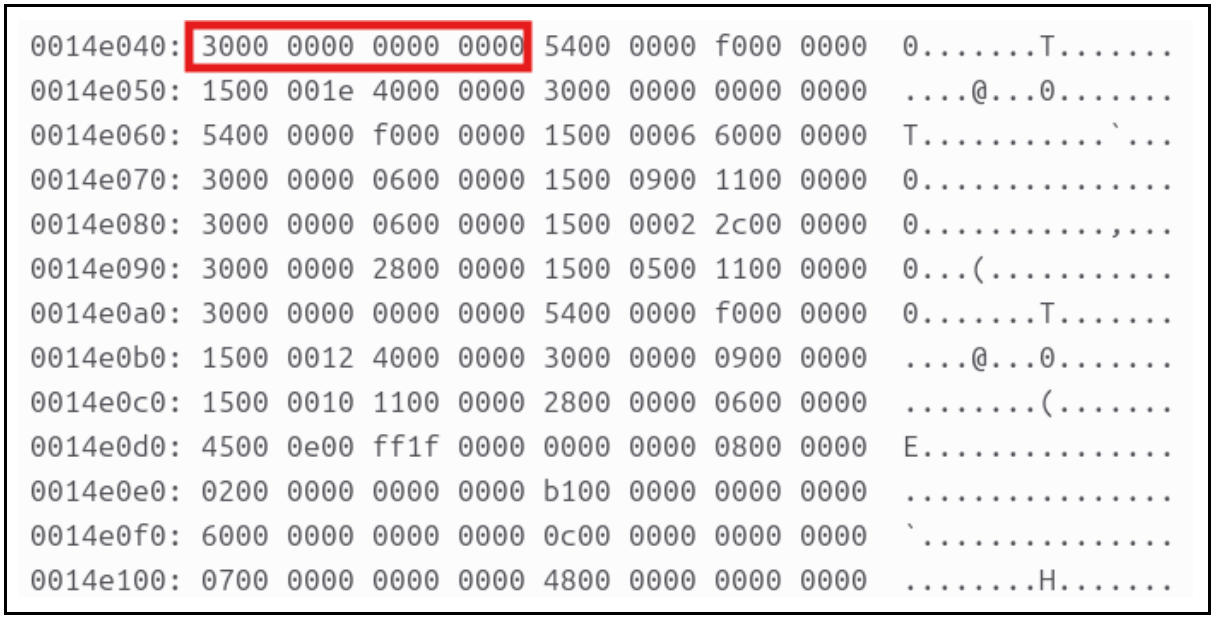

그림 32. hald-addon-volume xxd 출력 중 BPF 필터 시작값

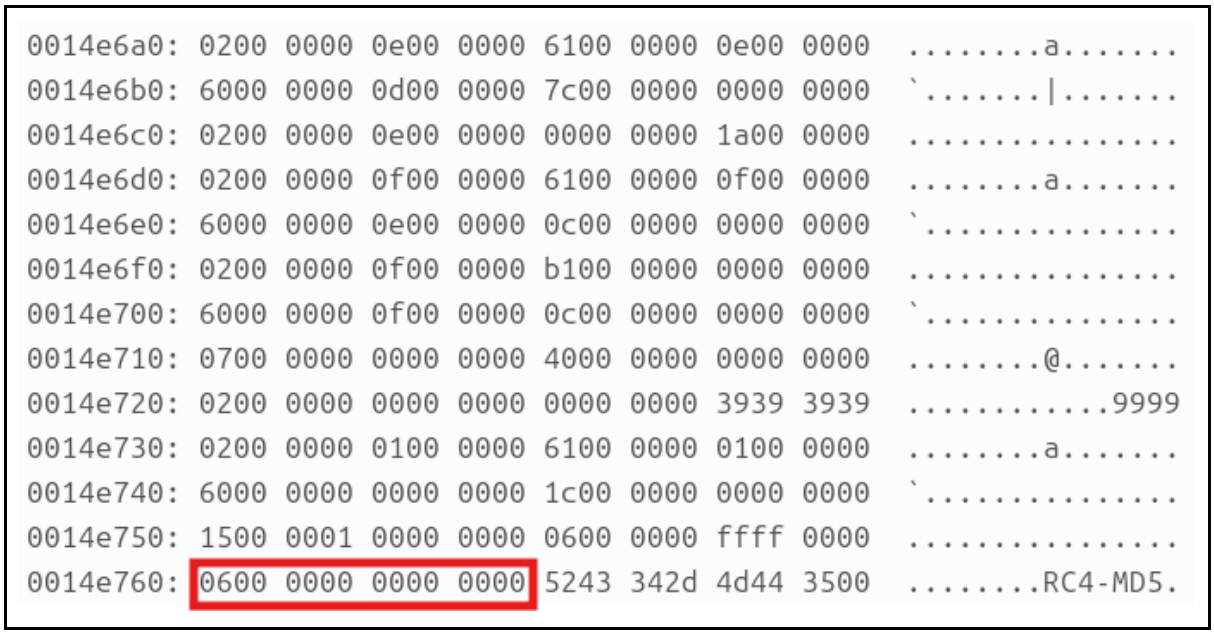

그림 33. hald-addon-volume xxd 출력 중 BPF 필터 마지막값

[그림 32,33]는 xxd 명령어를 통해 확인한 BPF 필터이다. 주소값 0x14e040 부터 0x14e768 까지 BPF 필터가 위치하는 곳으로 0x728 크기만큼을 차지하고 있다. BPF 필터에서 명령어 1개 당 8바이트이므로 0x728을 0x8로 나눈 값 0xe5는 10진수로 229이다. 따라서 229개의 명령어가 존재하는 필터로 D 유형 변종임을 확인할 수 있다.

6. smartadm (D 유형 변종)

다음은 SKT 침해사고에서 사용된 BPFDoor D 유형 변종인 smartadm 악성코드 실행 후 감염된 시스템에 남는 흔적 분석을 통한 악성코드 행위에 대한 분석이다.

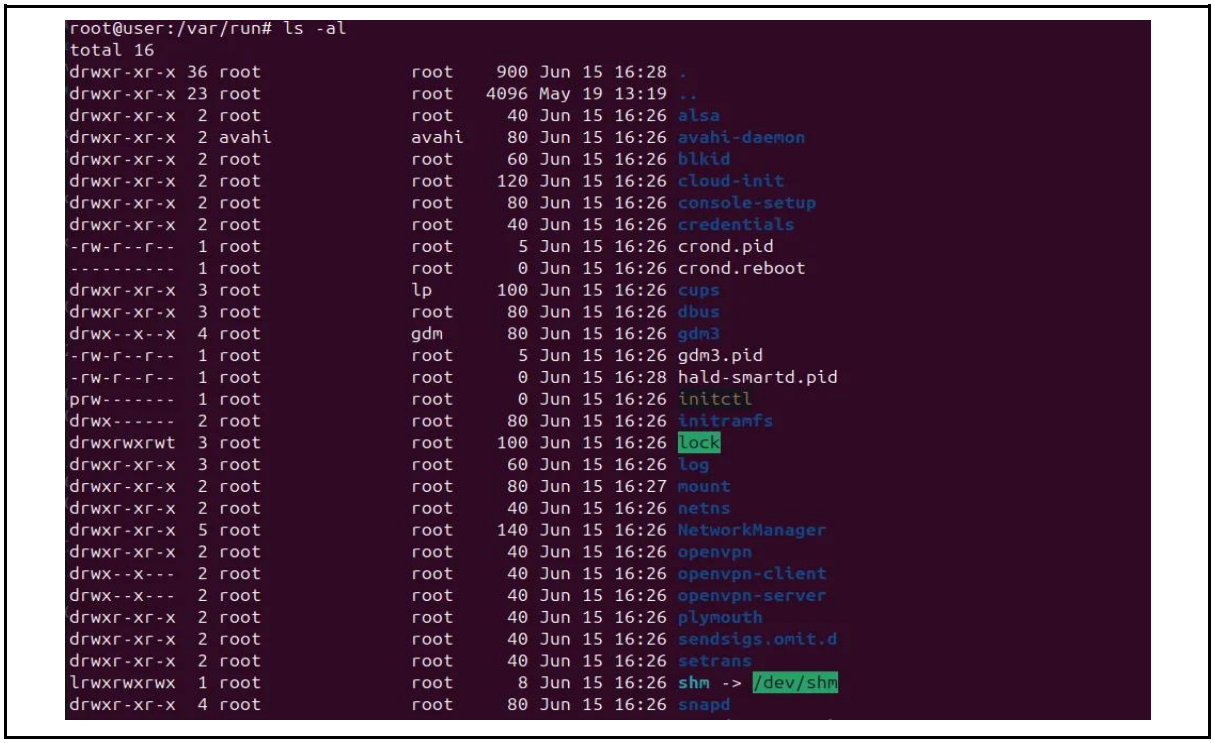

그림 34. smartadm mutex lock pid 명

악성 코드가 실행되면 hald-smartd.pid 파일을 생성하여 중복 실행을 방지한다.

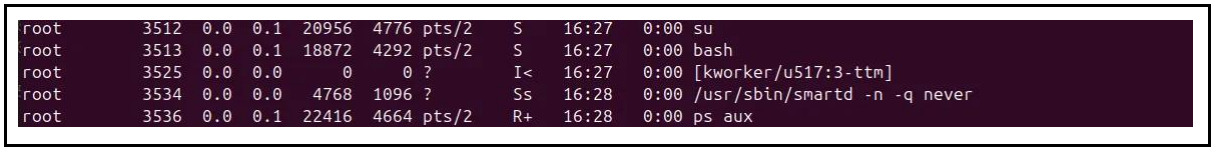

그림 35. smartadm 위장 프로세스명

pid가 3534인 프로세스명이 /usr/sbin/smartd -n -q never로 변경되었다.

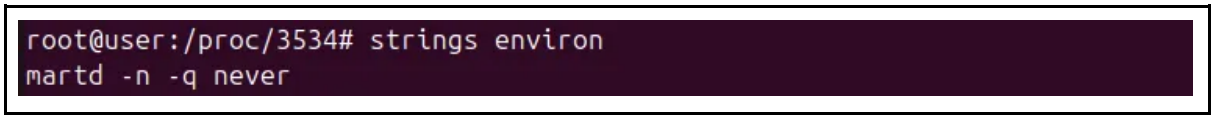

그림 36. smartadm 환경변수

프로세스명이 변경된 흔적을 남기지 않기 위해 기존 환경변수를 삭제하였다.

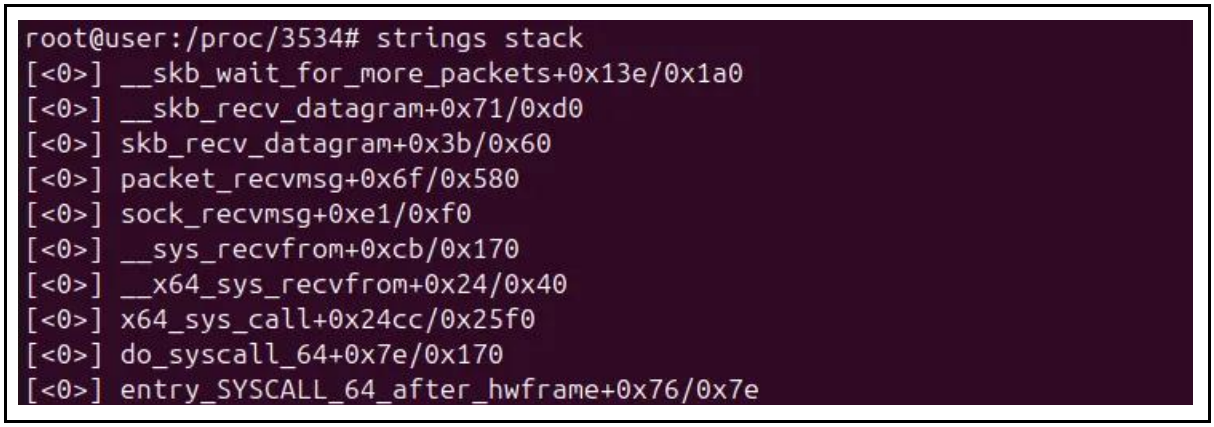

그림 37. smartadm 프로세스 stack

해당 프로세스의 /proc/[pid]/stack 정보를 통해 리스닝 상태 설정 및 수신된 패킷을 처리하기 위한 커널 함수가 들어있음을 확인할 수 있다.

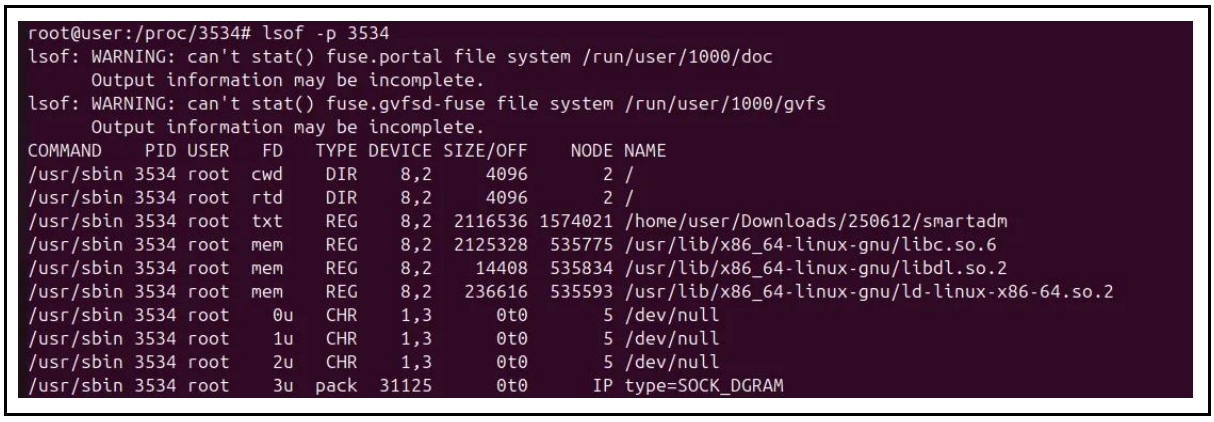

그림 38. smartadm 프로세스 lsof

해당 프로세스에서 열려있는 파일들을 통해 자가 복제 및 삭제 기능이 없기 때문에 삭제 흔적이 없고 SOCK_DGRAM이 열려있다. SOCK_DGRAM은 비연결 메시지인 데이터그램을 제공하며 UDP/IP 프로토콜에서 구현된다.

그림 39. smartadm 프로세스 파일 디스크립터

그림 40. smartadm 프로세스 소켓

31125번 소켓을 사용하고 있으며 type이 2로 SOCK_DGRAM이라는 것을 알 수 있다.

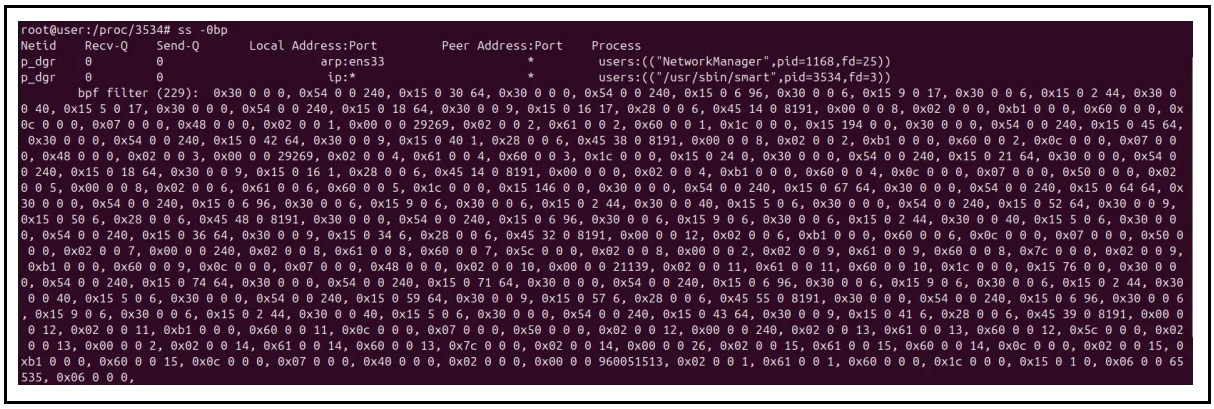

그림 41. smartadm 프로세스 매직 필터 설정 값

ss -0bp를 통해 229개의 명령어로 이루어진 BPF 필터이다. 이를 통해 BPFDoor D 유형 변종임을 파악할 수 있다.

7. dbus-srv-bin.txt (A 유형 변종)

다음은 SKT 침해사고에서 사용된 BPFDoor A 유형 변종인 dbus-srv-bin.txt 악성코드 실행 후 감염된 시스템에 남는 흔적 분석을 통한 악성코드 행위에 대한 분석이다.

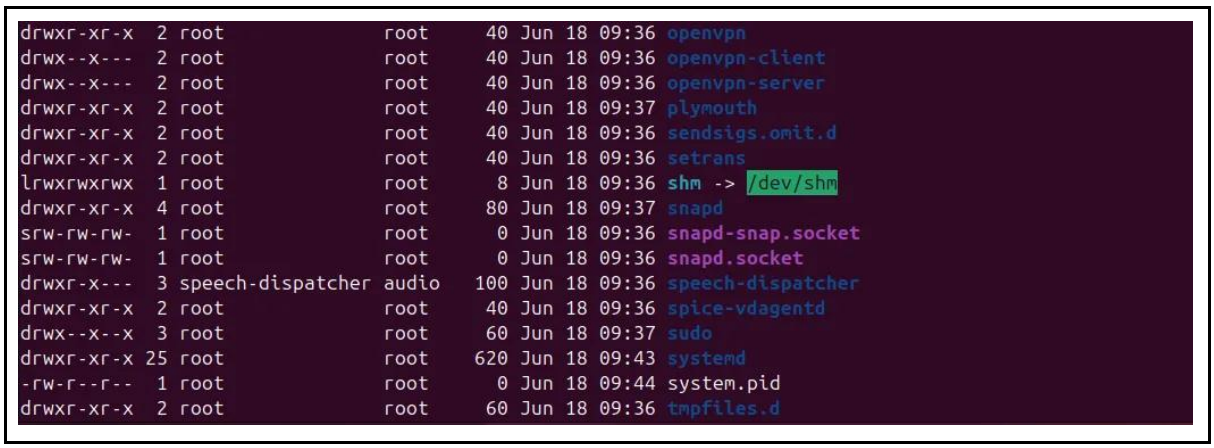

그림 42. dbus-srv-bin.txt mutex lock pid 명

system.pid 파일을 생성을 통해 mutex를 획득한다.

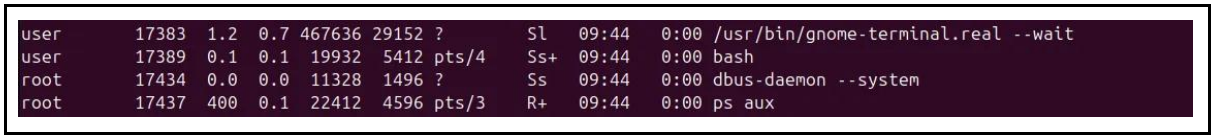

그림 43. dbus-srv-bin.txt 위장 프로세스명

dbus-daemon --system(pid: 17434)으로 프로세스명이 변경되었다.

그림 44. dbus-srv-bin.txt 환경변수

해당 프로세스의 환경변수를 삭제함으로써 추적을 막기위해 정보를 지워 흔적을 숨긴다.

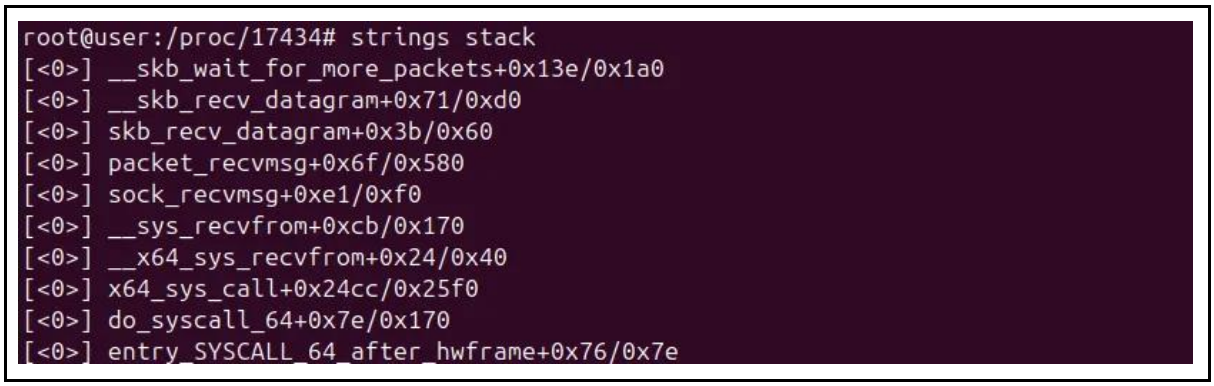

그림 45. dbus-srv-bin.txt 프로세스 stack

소켓 수신 관련 함수를 통해 pid 17434 프로세스가 패킷 수신 대기 상태에 있음을 알 수 있다.

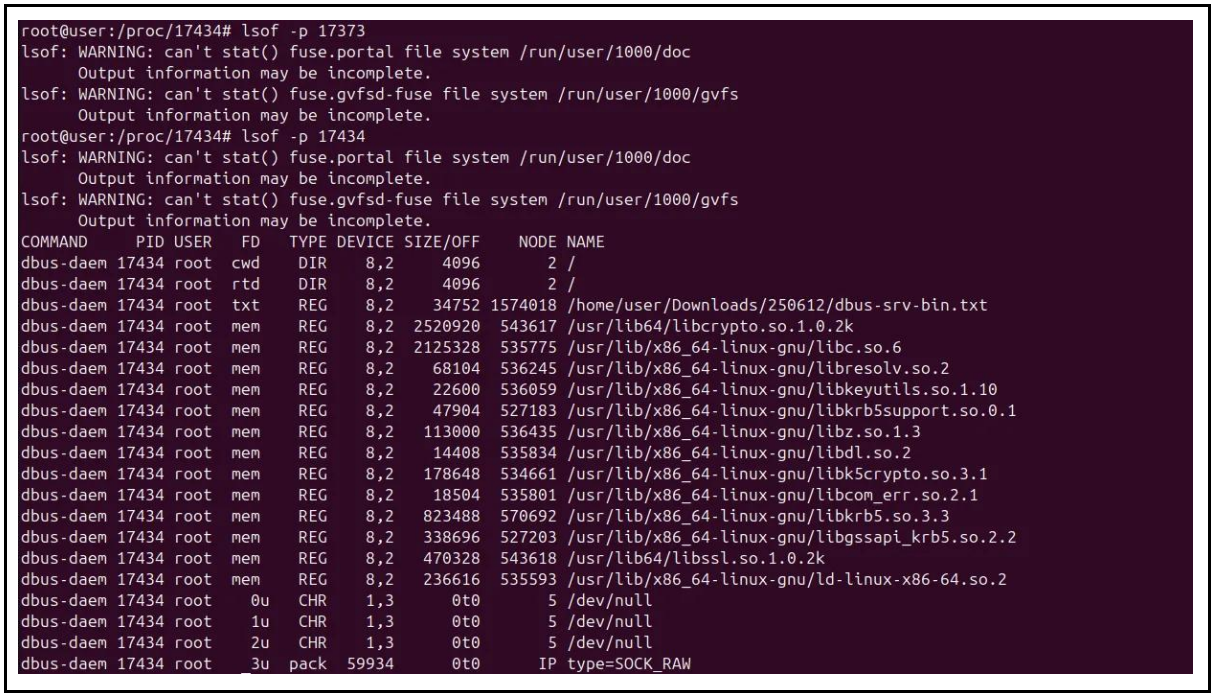

그림 46. dbus-srv-bin.txt 프로세스 lsof

lsof를 통해 해당 프로세스에서 열려있는 파일 확인으로 raw socket이 열린 것을 확인할 수 있다.

그림 47. dbus-srv-bin.txt 프로세스 파일 디스크립터

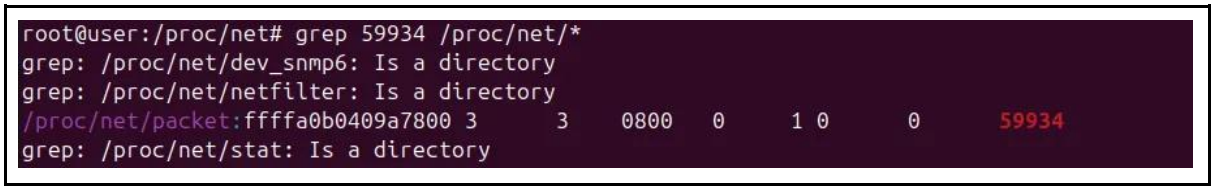

그림 48. dbus-srv-bin.txt 프로세스 소켓

59934번 소켓을 사용하며 이는 type이 3로 SOCK_RAW를 사용한다.

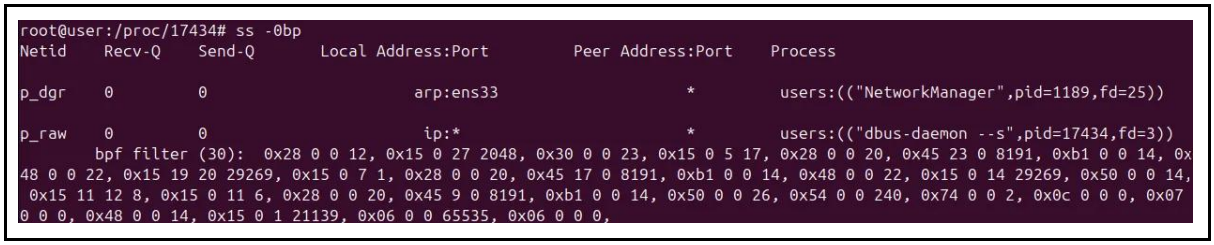

그림 49. dbus-srv-bin.txt 프로세스 매직 필터 설정 값

30개의 명령어로 이루어진 BPF 필터를 확인 가능하며 이는 A 유형의 변종이다.

8. 기존 BPFDoor A 유형 대비 4종의 변종 비교표

	기존 BPFDoor A 유형	hpasmmld	hald-addon-volume	smartadm	dbus-srv-bin.txt
명령 인증	단순 비밀번호 기반(justforfun, socket)	salt 값(I5*AYbs@LdaWbsO)을 이용한 MD5			
위장 프로세스명	/sbin/udevd -d /sbin/mingetty /dev/tty7 /usr/sbin/console-kit-daemon --no-daemon 등	hpasmlited -f /dev/hpilo	hald-addon-volume	smartd -n -q never	dbus-daemon --system
mutex lock	/var/run/haldrund.pid	/var/run/hp-health.pid	/var/run/hald-addon.pid	/var/run/hald-smartd.pid	/var/run/system.pid
암호화 통신	RC4	SSL 암호화(RC4-MD5)			
자가 복제, 실행, 삭제	/dev/shm/kdmtmpflush 이름으로 복제 및 실행 후 파일 삭제	자가 복제 및 삭제 기능 X			
BPF 필터	30개의 명령어	30개의 명령어	229개의 명령어	229개의 명령어	30개의 명령어

표 5. 변종 비교표

9. 결론

본 보고서를 통해 기존에 공개된 오픈소스 BPFDoor를 바탕으로 다양한 변종에 대한 분석을 진행하였다. BPFDoor는 BPF 필터를 기반으로 A~E 유형까지 나누어지며 변종들은 기존의 특성을 유지하면서도 명령 인증 방식, 위장 프로세스명, 복제 및 삭제 기능 제거, 그리고 SSL 암호화에서 차이를 보여 기존 유형에서 벗어나 더 다양하고 정교하게 변경되었음을 알 수 있었다.
특히 각 유형 간의 유사점과 차이점을 비교함으로써 공격자들이 탐지를 회피하고 지속성을 확보하기 위해 구조적 요소를 어떻게 조정했는지 구체적으로 파악할 수 있었다. 이를 통해 향후 유사 위협에 대응하기 위한 탐지 룰과 방어 전략 수립에 실질적인 기초 자료로 활용될 수 있을 것으로 보인다.

또한, 분석 과정에서 확인된 공통적인 패턴과 실행 방식은 BPF 기반 백도어가 가진 고유한 행위 기반 특징을 식별하는 데 도움이 되었으며 단순 시그니처 기반 탐지 방식으로는 식별이 어려운 변종들에 대해서도 효과적인 대응이 가능함을 시사한다.

BPFDoor는 여전히 진화 중이며 공개된 분석 정보가 공격자에 의해 역으로 악용될 수 있는 만큼 향후에는 실시간 탐지 및 방어를 고려한 다계층 분석 기반 접근이 병행되어야 한다. 본 보고서에서 제시한 변종 분석 및 유형별 비교 결과는 그러한 노력의 기반 자료로 활용될 수 있을 것으로 기대된다.

10. 별첨1 - IOC(**Indicator of Compromise; 침해 지표**)

		MD5	**SHA256**
hpasmmld	hpasmmld	a47d96ffe446a431a46a3ea3d1ab4d6e	c7f693f7f85b01a8c0e561bd369845f40bff423b0743c7aa0f4c323d9133b5d4
	hp-health.pid	d41d8cd98f00b204e9800998ecf8427e	e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
	hapsmlited -f /dev/hpilo	a47d96ffe446a431a46a3ea3d1ab4d6e	c7f693f7f85b01a8c0e561bd369845f40bff423b0743c7aa0f4c323d9133b5d4
hald-addon-volume	hald-addon-volume	f4ae0f1204e25a17b2adbbab838097bd	95fd8a70c4b18a9a669fec6eb82dac0ba6a9236ac42a5ecde270330b66f51595
	hald-addon.pid	d41d8cd98f00b204e9800998ecf8427e	e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
	/usr/libexec/hald-addon-volume	f4ae0f1204e25a17b2adbbab838097bd	95fd8a70c4b18a9a669fec6eb82dac0ba6a9236ac42a5ecde270330b66f51595
smartadm	smartadm	227fa46cf2a4517aa1870a011c79eb54	3f6f108db37d18519f47c5e4182e5e33cc795564f286ae770aa033

			72133d15c4
	hald-smartd.pid	d41d8cd98f00b204e9800998ecf8427e	e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
	/usr/sbin/smartd -n -q never	227fa46cf2a4517aa1870a011c79eb54	3f6f108db37d18519f47c5e4182e5e33cc795564f286ae770aa03372133d15c4
dbus-srv-bin.txt	dbus-srv-bin.txt	714165b06a462c9ed3d145bc56054566	aa779e83ff5271d3f2d270eaed16751a109eb722fca61465d86317e03bbf49e4
	system.pid	d41d8cd98f00b204e9800998ecf8427e	e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
	dbus-daemon --system	714165b06a462c9ed3d145bc56054566	aa779e83ff5271d3f2d270eaed16751a109eb722fca61465d86317e03bbf49e4

표 6. IOC 표

11. 별첨2 - 유형별 BPF 필터

유형 A

MD5	549736782cbf728cfdc21936f61c5f85
SHA256	f9b762d23b8fcb14a09a5f8178191a5290a378ca42c248ef50f3e4abb326550a

```
root@user:/proc/14081# ss -0bp
Netid  Recv-Q   Send-Q     Local Address:Port      Peer Address:Port  Process
p_dgr  0        0                  arp:ens33               *       users:(("NetworkManager",pid=1242,fd=26))
p_raw  0        0                   ip:*                   *       users:(("/usr/sbin/conso",pid=14081,fd=3))
       bpf filter (30):  0x28 0 0 12, 0x15 0 27 2048, 0x30 0 0 23, 0x15 0 5 17, 0x28 0 0 20, 0x45 23 0 8191, 0xb1 0 0 14
, 0x48 0 0 22, 0x15 19 20 29269, 0x15 0 7 1, 0x28 0 0 20, 0x45 17 0 8191, 0xb1 0 0 14, 0x48 0 0 22, 0x15 0 14 29269, 0x50
 0 0 14, 0x15 11 12 8, 0x15 0 11 6, 0x28 0 0 20, 0x45 9 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 0 1 21139, 0x06 0 0 65535, 0x06 0 0 0,
```

그림 50. A 유형 BPF 필터

유형 B

MD5	acea44892fc67223f43f4af2ec81aa83
SHA256	d68948964905af7259bca015bd1d1ab0bb54334a6f08a87a40ed9d8cc966b291

```
root@user:/proc/35184# ss -0bp
Netid     Recv-Q     Send-Q        Local Address:Port         Peer Address:Port    Process

p_dgr     0          0                    arp:ens33                   *         users:(("NetworkManager",pid=1140,fd=26))

p_raw     0          0                     ip:*                       *         users:(("dbus-daemon --s",pid=35184,fd=3))
       bpf filter (39):  0x28 0 0 12, 0x15 0 36 2048, 0x30 0 0 23, 0x15 0 5 17, 0x28 0 0 20, 0x45 32 0 8191, 0xb1 0 0 14, 0x48
0 0 22, 0x15 28 29 29269, 0x15 0 7 1, 0x28 0 0 20, 0x45 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 20 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, 0
x48 0 0 14, 0x15 9 0 21139, 0xb1 0 0 14, 0x50 0 0 26, 0x54 0 0 240, 0x74 0 0 2, 0x04 0 0 26, 0x0c 0 0 0, 0x07 0 0 0, 0x40 0 0 14
```

그림 51. B 유형 BPF 필터

유형 C

MD5	0bf4d063ffe48c5419a1d68f58e99815
SHA256	a9884ba3ebf25dcb1b9b3319d5e9e3706832bfa0f1fc4248f22a065f7ef15f79

TCP	0x5293 0x39393939	총 229개의 명령어가 포함
UDP	0x7255	
ICMP	0x7255	

표 7. C 유형 BPF 필터

※ 위 해시 값은 Trendmicro[2]의 IOC에 따른 C 유형이며 바이너리는 공개되지 않았다. 따라서, S2W에서 밝힌 C 유형의 BPF 필터에 적용된 매직 넘버 값을 첨부한다[3].

유형 D

MD5	227fa46cf2a4517aa1870a011c79eb54
SHA256	3f6f108db37d18519f47c5e4182e5e33cc795564f286ae770aa03372133d15c4

그림 52. D 유형 BPF 필터

유형 E

MD5	0cd3b5acfab2d6081a2cb48c4c711fd3
SHA256	afa8a32ec29a31f152ba20a30eb483520fe50f2dce6c9aa9135d88f7c9c511d7

그림 53. E 유형 BPF 필터

12. 참고문헌

[1] KISA, “최근 해킹공격에 악용된 악성코드, IP 등 위협정보 공유 및 주의 안내”, https://www.boho.or.kr/kr/bbs/view.do?bbsId=B0000133&pageIndex=1&nttId=71726&menuNo=205020

[2] Trendmicro, Detecting BPFDoor Backdoor Variants Abusing BPF Filters, https://www.trendmicro.com/en_us/research/23/g/detecting-bpfdoor-backdoor-variants-abusing-bpf-filters.html

[3] S2W, “Detailed Analysis of BPFDoor targeting South Korean Company”, https://medium.com/s2wblog/detailed-analysis-of-bpfdoor-targeting-south-korean-company-328171880a98

[4] Anlab, BPFDoor 악성코드 분석 및 안랩 대응 현황, https://www.ahnlab.com/ko/contents/content-center/35830

[5] haxrob, BPFDoor Part 2 – The Present, https://haxrob.net/bpfdoor-past-and-present-part-2/