

[www.halfdog.net / Security / 2015 / ApportKernelCrashdumpFileAccessVulnerabilities /](http://www.halfdog.net/Security/2015/ApportKernelCrashdumpFileAccessVulnerabilities/)

Share via [f](#) [g+](#)

Introduction

Problem description: On Ubuntu Vivid Linux distribution *apport* is used for automated sending of client program crash dumps but also of kernel crash dumps. For kernel crashes, upstart or SysV init invokes the program */usr/share/apport/kernel_crashdump* at boot to prepare crash dump files for sending. This action is performed with root privileges. As the crash dump directory */var/crash/* is world writable and *kernel_crashdump* performs file access in unsafe manner, any local user may trigger a denial of service or escalate to root privileges. If symlink and hardlink protection is enabled (which should be the default for any modern system), only denial of service is possible.

Problematic syscall in *kernel_crashdump* is:

```
open("/var/crash/linux-image-3.19.0-18-generic.0.crash", O_WRONLY|O_CREAT|O_TRUNC|O_LARGEFILE|O_CLOEXEC, 0666) = 30
...
open("/var/crash/vmcore.log", O_RDONLY|O_LARGEFILE|O_CLOEXEC) = 31
```

Thus the output file is opened unconditionally and without *O_EXCL* or *O_NOFOLLOW*. Also opening of input file does not care about links.

Methods

Denial of service: It is trivial to make the input and output file the same by just placing a hardlink. This is allowed even with hardlink protection enabled as the unprivileged user doing that will have write permissions for the rogue file and the new hardlink pointing to it. On reboot *kernel_crashdump* will include its own output over and over again in the crashdump till the disk is completely filled. This can be done using [PrepareDoSBoot.py](#).

Privilege escalation: With symlink or hardlink protection turned of, [PrepareMagicBoot.py](#) can be used to escalate privileges. While the link attack is really trivial (see code), the method to gain full code control is quite nice: The problem is, that *kernel_crashdump* creates following report format:

```
ProblemType: KernelCrash
Architecture: i386
Date: Fri Sep  4 20:36:21 2015
DistroRelease: Ubuntu 15.04
Package: linux-image-3.19.0-26-generic 3.19.0-26.28
Uname: Linux 3.19.0-26-generic i686
VmCore: base64
H4sICAAAAAAC/1ZtQ29yZQA=
AwA=
VmCoreLog: base64
H4sICAAAAAAC/1ZtQ29yZUxvZwA=
AwA=
```

The structure contains some predefined fields and compressed base64 encoded log data and core. Overwriting */etc/shadow* (line structure violated) or */etc/ld.so.conf.d/libc.conf* (requires absolute path) with that content would just cause DoS, so not so interesting. Overwriting executables does not help neither as on valid execution file header can be produced. But there are scripts, e.g. */etc/rc.local*, that execute text without any need for a header in their context, e.g. */lib/init/vars.sh*. Overwriting those would also just result in errors, because none of the included attribute names is user controllable to make it refer to a path the unprivileged local user could control. So the last chance is to make the base64 encoded, compressed log or core file data to point to a suitable location.

Generation of such data is not as simple as it seems. To avoid having additional requirements on available directories on that machine, only existence of */var/crash* should be needed, no attempts should be made to construct data for */home* or */var/www/...* As */etc/rc.local* is executed with current working directory at */*, also relative path would be sufficient.

Due to *deflate compression* method used before encoding, there are only some combinations of lower bits in the first byte allowed. See section *details of block format* in [RFC1951](#). As a consequence, it is not possible to create compressed data that would start with *var/crash* (last chunk in dynamic encoding, but literal/distance/code length codes do not form a valid combination), */var/crash* (non-last chunk in reserved encoding) of *//var/crash* (last chunk in reserved encoding). Hence no compressed data block will start with suitable values, but code analysis reveals another chance:

kernel_crashdump will split long dumps into blocks of 1MB size, compress them and write the base64 encoded output to separate lines. This is especially interesting as deflate operates on bit basis, compression output is in bytes. Hence when one block is compressed, some bits remain in the output buffer and are combined with bits from the next block. Thus it might be possible to find two adjacent blocks where the bits at this position may form the desired sequence of bytes.

As detail analysis of the deflate algorithm to carefully craft a suitable first block was too much effort, a program to guess appropriate blocks was used ([CreateVmcore.py](#)). Invoking it to target *var/crash* sequence and alike did not yield any results.

Also strings with additional slashes prepended or within path could not be created. Luckily, one alternative path was

found, that should exist on all systems and can be the output of the compression and encoding transformation: `proc//1/root/var/crash/`.

```
$ ./CreateVmcore.py proc//1/root/var/crash/A
...
58045-Tail ec-7Nb5XxsEvuh9 (45983) last 880 b6366ec99f8f (tjZuyZ+PHP65)
Good result: proc//1/root/var/crash/AkrxiSugWCRV9CZ0GL...8TAQ==
```

Just replacing some bytes with 0 and modifying the tail to be shorter not to exceed the maximum path length. For exploitation following steps were performed:

- Copy `/lib/init/vars.sh` to `/var/crash`
- Prepare vmcore and symlink to overwrite `/lib/init/vars.sh` with crash dump report
- Wait for reboot and invocation of `/etc/rc.local`
- Restore `/lib/init/vars.sh` to be back to normal

[PrepareMagicBoot.py](#) performs all the steps to prepare the system for reboot. It also includes the vmcore data to be compressed by the `kernel_crashdump`.

```
$ ./PrepareMagicBoot.py
nobody@localhost:/var/crash$ ls -al
total 1068
drwxrwxrwt 2 root root 4096 Sep 5 09:27 .
drwxr-xr-x 13 root root 4096 May 12 11:23 ..
-rwxr-xr-x 1 nobody nogroup 99 Sep 5 09:27 AuttsThCkBCKV1JSVRkcgRsEooBsAAHSEdI0AABAA
lrwxrwxrwx 1 nobody nogroup 17 Sep 5 09:27 linux-image-3.19.0-26-generic.0.crash -> /lib/init/vars.sh
-rwxr-xr-x 1 nobody nogroup 19883 Sep 5 08:14 PrepareMagicBoot.py
-rw-r--r-- 1 nobody nogroup 1212 Nov 10 2014 vars.sh
-rw-r--r-- 1 nobody nogroup 1048576 Sep 5 09:27 vmcore
-rw-r--r-- 1 nobody nogroup 0 Sep 5 09:27 vmcore.log
```

Results, Discussion

Not a big issue, as symlink protection should be standard nowadays. But the crafting of the core file was fun, so let others enjoy also.

Timeline

- 20150905: Report at Ubuntu Launchpad: [1492570](#)
- 20150917: CVE assigned: [CVE-2015-1338](#)
- 20150924: Updates released by Ubuntu, see [USN-2744-1](#)
- 20150924: Full disclosure post of this article, see [2015/Sep/101](#)

Material, References

- Tools: [PrepareDoSBoot.py](#), [PrepareMagicBoot.py](#), [CreateVmcore.py](#)
- Deflate RFC: [RFC 1951](#)
- Ubuntu Launchpad Bug Report: [1492570](#)
- Mitre CVE record: [CVE-2015-1338](#)

Last modified 20150927
Contact e-mail: me (%) halfdog.net